Drucksachenkategorie

Watertight Surface Reconstruction for Uncertain Data

Markus Wiedemann

DLR

Institut für Robotik und Mechatronik

BJ.: 2014

IB.Nr.: 572-2014/30

MASTER'S THESIS

WATERTIGHT SURFACE RECONSTRUCTION FOR UNCERTAIN DATA

Freigabe:	Der Bearbeiter:	Unterschriften
	Markus Wiedemann	
	Betreuer:	
	Simon Kriegel	
	Der Institutsdirektor	
	Prof. Dr. Alin Albu-Schäffer	

Dieser Bericht enthält 73 Seiten, 54 Abbildungen und 2 Tabellen

Ort: Oberpfaffenhofen	Datum: 03.12.2014	Bearbeiter: Markus Wiedemann	Zeichen:

Watertight Surface Reconstruction for Uncertain Data

eingereichte
MASTERARBEIT
von

cand. ing. Markus Wiedemann

geb. am 27.07.1988
wohnhaft in:
Lechstraße 10
86931 Prittriching
Tel.: 0172 8557139

Lehrstuhl für
INFORMATIONSTECHNISCHE REGELUNG
Technische Universität München

Univ.-Prof. Dr.-Ing. Sandra Hirche

Betreuer:	Dipl.-Ing. Simon Kriegel (DLR), Dr.-Ing. M. Leibold
Beginn:	01.05.2014
Zwischenbericht:	15.08.2014
Abgabe:	02.12.2014

In your final hardback copy, replace this page with the signed exercise sheet.

Abstract

For simple pick and place tasks, a robot needs to be able to recognise objects. In order to avoid scanning all objects by hand, an automatic modelling approach is necessary. Considering space and price restrictions on a robotic platform, the quality of perceiving devices suffers from this restrictions. Consequently modelling approaches need to cope with pose errors and noisy data that usually result in displaced surfaces incorporating a lot of artefacts.
In this work, we focus on creating a watertight surface representation that is automatically built by a robot. Therefore, we develop a next best viewpoint planning method that fits to an afterwards applied filtering stage for improving measurement confidence. Using the resulting data, a mesh growing approach reconstructs the surface of the to be scanned object incorporating an inflating and a detailing stage. Our approach is able to roughly estimate the objects surfaces using inaccurate hardware and even to reconstruct small details for precise laser scanning devices. Small areas that are not scanned can be filled and highlighted by utilizing the near neighbourhood of measurements.

Zusammenfassung

Damit Hol-und-Bring Aufgaben ausgefhrt werden knnen, muss ein Roboter die zu bringenden Objekte erkennen. Um zu vermeiden, dass all dieses Objekte per Hand eingescannt werden, ist eine Applikation notwendig, die dies automatisch durchfhrt. Durch beschrnktes Platzangebot und einen eingeschrnkten Kostenrahmen einer robotischen Plattform leidet die Qualitt und Genauigkeit von optischen Messgerten.
Das fhrt dazu, das Modell bildende Anstze Fehler in Pose und verrauschte Daten, die zu versetzten und mit Artefakten besetzten Oberflchen fhren, beachten mssen. In dieser Arbeit konzentrieren wir uns auf einen Ansatz, der eine wasserdichte Oberflchenreprsentation automatisch mithilfe eines Roboters erstellt. Dafr entwickeln wir zunchst eine Planungsphase, die den nchste Scanpunkt generiert und dabei die Anforderungen der darauf folgenden Filterphase erfllt. Die gefilterten Daten werden dann von einem Ansatz benutzt, der ein Gitternetz wachsen lsst, das die Oberflche eines zu scannenden Objektes darstellt. Jener besteht aus einer Inflations- und einer Detaillierungsphase.
Unsere Applikation ermglicht im Angesicht von ungenauer Hardware eine grobe Schtzung von Oberflchen und bei Benutzung von hoch przisen Laser Scannern, sogar die Rekonstruktion von kleinen Details. Kleine Bereiche, die gescannt wurden, knnen hervorgehoben und rekonstruiert werden, indem rtlich nahe liegende Messungen benutzt werden.

Contents

Chapter 1

Introduction

In recent years, the field of human assisting service robots is getting more and more attention in robotic research. For robots assisting humans, it is necessary to enable them to perceive their environment and create a computer understandable representation of it. This work focuses on the generation of a mesh based on scan data utilizing a mobile robot equipped with a manipulator, to scan given objects and create a mesh representation of them.

1.1 Problem Statement

For scanning objects, a variety of sensors exist that scan the distance to a surface. Examples for this are laser stripers that come with a high price but have the advantage of precise measurement. In comparison to that, another popular and cheap example is Microsoft's Kinect camera. However, it is not able to reach the accuracy of a laser striper. For creating mesh representations, those drawbacks have to be kept in mind.
Due to discretisation and measuring failures, a straight forward meshing algorithm would produce discontinuous and self-intersecting representations of real world objects. A further error occurs hand in hand with a position tracking device. This can be cameras in combinations with markers or a manipulator that either holds an object or a camera mounted to it. Those pose errors lead to displaced point clouds and therefore failures in the depth perception. Another source of error are reflections by translucent media. They induce further errors in the measured depth. Consequently, appropriate methods have to be found and implemented to cope with uncertainties of 3D perception.

1.2 Related Work

For surface reconstruction it is first necessary to plan the scanning of an unknown object. Therefore, viewpoints need to be generated. This means the position and

the viewing direction of a scanning device need to be calculated. Related work on planning the next best view will be introduced in the succeeding subsection. After scanning, an automatic algorithm is expected to cope with errors in orientation and position of the scanning device. Considering those errors to be only small in the given scenario, we will take a closer look at the Iterative Closest Point (ICP) algorithm. Finally, we introduce the basic concept of deformable surface reconstruction that aims at building a mesh consisting of triangles in order to represent the surface of the scanned object.

1.2.1 Next Best View Planning

The first step in scanning real world objects is to plan the Next Best View (NBV). Here, suitable positions and view directions of the scanning device have to be generated and evaluated with respect to their information gain. There exists a variety of solutions for this problem. Lamp et al. [1] implemented an automatic scanning system where five strategies are implemented that are described as follows:

Edge-Scan Edges are defined as being the connecting between two faces with different normals. The scan is oriented towards the negative mean of the two normals and follows the edge.

Raster-Scan Here, the scan is directed on a planar surface and oriented along its negative normal.

Box-Scan For the Box-Scan, the scan performs the Edge-Scan and a Raster-Scan on a the top edges and five faces of a virtual box.

Profile-Scan Using a previously performed scan, here the scanner follows the contour of a raw model.

Servo-in-Depth Here, the scanner's trajectory follows the contour of the model by using online the already scanned surface.

The authors aimed at implementing a semi-automatic scanning software where the user decides which scan strategy is to be used.
A further strategy based on a mesh model was introduced by Khalfaoui et al. [2]. They calculate the Mass Vector Sum (MVS) - introduced by Loriot et al. [3] - by adding up the surface normals of the boundary patches of the model. View point candidates are produced by utilizing the surface normals of nearby surfaces of the MVS. For each view point candidate k, a weight in timestep i is calculated by

$$w_k^i = \frac{\theta_k^i}{\theta_{max}^i}$$

where θ_{max}^i is the maximum difference angle and θ_k^i the difference angle of the current viewpoint direction to the MVS. All viewpoints are then grouped in clusters,

the average weight for each cluster is calculated and the cluster with the highest average weight is chosen as next best view point.
Massios and Fisher [4] generate viewpoints on a sphere around the to be scanned object. They first filter the viewpoints with respect to their reachability due to kinematic restrictions of the robot. The remaining viewpoints then are evaluated on a voxelspace by a visibility and a quality criteria. The visibility criteria represents the number of visible voxels that were occluded in previous scans. For the quality criteria all previously scanned voxels are used. The quality q_i of each voxel is estimated for each performed scan and used for the next. It is calculated by averaging the normals of all surfaces inside this voxel and calculating the dot product of the average normal and the viewing direction. The quality of a voxel is only changed, if the current scan's quality is higher than the previous ones. This quality is then used for calculating a further quality value $f_{quality}$ for a simulated scan that scans the set of M voxels:

$$f_{quality} = \frac{1}{|M|}\left(\sum_{i=1}^{M}(1-q_i)(|v \cdot n_i)\right)$$

where v is the viewing direction and n_i the average normal of voxel i. Both criteria are multiplied by individual weights and then added up to a total quality estimation value.
Kriegel et al. [5] estimate the curvature of the boundaries of the mesh model. Based on that viewpoints are generated perpendicular to the estimated curvature. They introduce in [6] a quality criteria for determining the next best view point. Therefore, the authors estimate the total information gain by performing a simulated scan for each viewpoint. Using a voxelspace whose states describe the probability p of a voxel being occupied, for each by the simulated scan intersected voxel a entropy H_{voxel} is calculated as

$$H_{voxel} = -p\log(p) - (1-p)\log(1-p).$$

By adding up H_{voxel} for all intersected voxels, the information gain for a certain viewpoint can be estimated. Furthermore, they introduce in [7] an additional criteria for determining the next best view. They estimate the quality q_i of a surface patch of the voxel i by incorporating the point density $\bar{d}_i$ of that voxel and the proportion of boundaries inside it relative to the total number of boundaries b_i:

$$q_i = \lambda \cdot b_i + (1-\lambda) \cdot \bar{d}_i.$$

Here, λ is a user defined parameter for weighting the two criteria. Furthermore, if the average normal of all surfaces inside the voxel differs more than 70° from the viewing direction, q_i is set to zero for that voxel.

1.2.2 Iterative Closest Point

In recent past, many different variants of the ICP algorithm were published. The first versions were introduced by Chen and Medioni [8]in 1991 and Besl and McKay [9]

in 1992.
The main idea of ICP is to iteratively minimise the distance between two point clouds. Therefore, we first define one as the reference cloud which is fixed to its pose. The second cloud is iteratively aligned to the reference cloud and we refer to it as data cloud in the following. The variants of ICP differ in several aspects, therefore we structure the key points in the succeeding section in the style of [10].

Selection

For choosing which points of the data cloud to use multiple possibilities exist. Chen and Medioni [8] suggest to use regularly distributed points that are in a "smooth neighbourhood". In order to check this, the authors suggest to fit a smooth surface function, e.g. a plane, to the neighbourhood by using least squares and afterwards evaluating the standard deviation.
Besl and McKay [9] use all points of the data cloud. In contrast, Toldo et al. [11] use only those points of the data cloud that are the mutually closest points. This means, that we search in the reference cloud the closest points to every point of the data cloud. Mutually closest is, if this is as well vice versa.
Masuda et al. [12] use random sampling on the data cloud. Zhang [13] as well as Neugebauer [14] propose to use only a small number of points that are uniformly distributed until a certain convergence criterion is fulfilled. Then more and more points are included until all points are used. Neugebauer [14] supposes a speed-up factor of $20 - 100$. A dynamical selection of points was proposed by Chetverikov et al. [15]. They suppose to calculate the distances between each point pair and sorting them with respect to their distance. Then only a fixed number of them representing the smallest distances are used.

Correspondences

In order for aligning points to the data cloud, we need to select corresponding points in the reference cloud.

Point to Point Besl and McKay [9] use the closest point of the reference cloud for each point of the data cloud.
Using a k-d-tree can significantly accelerate this search as proposed by Simon [16].

Closest Line Another possibility was introduced in 2008 by Censi [17] that utilizes the point to line distance. For each point of the data cloud the closest two points of the reference cloud have to be found. Then the distance between the first point and the connecting line of the latter two points is minimized. This approach was introduced only for 2D data.

Point to Plane The chronologically first published correspondence selection approach is the point to plane projection introduced by Chen and Medioni [8]. It follows the work of Potesil [18] by using the surface normal of the selected point of the data cloud and intersecting it with the reference cloud. Therefore, the points of the reference cloud are approximated using planes and then the intersection of the surface normal with these planes are calculated.
Another possibility for the point to plane approach is to project the points of the data cloud onto the reference cloud in the direction of the camera view. This means that we first transform the data cloud in the coordinate system of the reference cloud. Then the depth value is adjusted to be aligned to the reference clouds surface. Afterwards, we search the nearest point on this surface. This idea was introduced in 1997 by Neugebauer [14].
Masuda et al. [12] use the point-to-triangle distance which is closely related to the point to plane distance. They as well optimise the search for the nearest triangle with the use of a k-d-tree.

There exist other possibilities to find correspondences, e.g. based on colour or light intensity which we do not list here, as we solely concentrate on the given depth information.

Correspondence Weighting

Once correspondences are found, we can weight them in different ways. The first two approaches by Besl and McKay as well as Chen and Medioni do not include any weighting strategies and therefore, this can be seen as using a constant weight of 1. Rusinkiewicz [10] considers using weighting based on the point-to-point distance for the correspondences. Here, points that are farther away from their correspondences get lower weights and vice versa. Another possibility examined by Rusinkiewicz and Levoy is to use the "compability of normals" expressed by

$$Weight = n_1 \cdot n_2$$

as well as to use the expected noise produced by the scanning device. This error metric is highly dependent on the used hardware and examples for different noise models utilized for depth measuring devices can be found in [19, chap. 3], [20] or [21]. Toldo et al. propose to compute the "Median of Absolute Deviations" MAD [11] and use its inverse value, meaning $1 - MAD$.
A further method was introduced in 2009 by Segal et al. [22]. They propose to use the property that a real world object has to be at least locally planar. Consequently, the probability distribution along the normal of a point coincides with the uncertainty of the measuring device. In the directions orthogonal to the normal the uncertainty should become bigger and therefore, the main movement of a point while registering should be applied to this directions.

Point Rejection

As the hardware used for obtaining point clouds are bound to noise and uncertainties, we have to consider the possibility of outliers. This means that there are points generated due to reflections or other sources of error. We need an efficient approach to deal with those outliers. Turk and Levoy [23] use a fixed threshold to discard outliers. Pandey et al. [24] define this threshold to be dynamically calculated. This is done by estimating the camera motion and restricting the correspondence search to a local region. Furthermore, Turk and Levoy discard correspondences that lie on a boundary of a mesh. Considering a surface meshed according to the techniques presented in sec. 1.2.3, if this surface ends somewhere, the points on this ending are not allowed to be correspondences for the ICP algorithm.
Masuda et al. [12] use a classification algorithm to determine whether the points of the data clouds are "inliers" or "outliers". Therefore, they check if the distance between points of the data cloud to the meshed model are within a threshold. If they are, they are inliers and are used for further calculation. This step is repeated in every iteration and therefore, inliers can become outliers and vice versa during the algorithm. The threshold used by Masuda et al. is 2.5 times the standard deviation. A similar approach is used by Zhang [13] but with a small deviation. His threshold is based on the relation between the old threshold and the current standard deviation and a first threshold has to be set by the user.
Pulli [25] suggests using multiple conditions. The first condition is to use only point pairs if their normals do not differ more than 45°. As second, the authors integrated the idea of Turk and Levoy [23] where points on the mesh boundaries are rejected. Furthermore, only a percentage of the closest point pairs are considered and therefore all other points are rejected. Additionally, correspondence pairs whose distance is greater than a certain threshold are rejected as well. The remaining pairs are updated in every step.
Using these techniques, there exists a variety of ICP algorithms. A small number of some of these variants or its derivatives can be found in [8–12, 15, 22, 24–27] and others.

1.2.3 Surface Reconstruction using Deformable Meshes

Using 2.5D cameras comes with the price of noisy data. However, for being able to reconstruct detailed surface representations of real world objects, we want to implement a deformable meshing algorithm. The advantages of this are that more than one scan can be integrated in a mesh and outliers or noisy data can be corrected by new measurements. We hope to be able to improve scanning, even though cheaper and consequently lower quality devices are utilized for perception.
The idea of deformable entities was first introduced by Kass et al. [28] in 1988. They described snakes as splines that are attracted to significant image features such as edges. This framework is designed to integrate user input as initial positions of the snakes, applying pushing forces to move them out of local minima and moving them smoothly with a spring dynamic. Their inner dynamic is constantly minimizing its energy and therefore is even able to track moving edges in intensity images.
Miller et al. [29] based on this idea a framework for extracting topology closed models for 3D data. They extended the snake approach for the use of 3D polygons in order to reconstruct meshes of real objects. In order to do so, they use a seed model as initial configuration, which is then expanded until the surface of an object is reached.
A further extension was introduced in 1992 by Vasilescu and Terzopoulos [30]. In their approach, a discontinuity detection was implemented which in case of found discontinuities could subdivide polygons, and merge them if necessary.
The next step towards state of the art deformable meshes was published in 1998 by Lachaud and Montanvert [31]. They introduced a generic approach that allowed to construct and deform a mesh in different resolutions. This means, their framework starts with a low resolution in order to build a coarse model of the object. In the further proceeding, the coarse model is refined by dividing the triangles iteratively up to the desired resolution or edge length. With this framework, the first self collision scheme was introduced. Here, not neighbouring triangles whose distance is below a certain threshold start a melting process between the two parts of the mesh.
In the following, we will give a brief description of the different node dynamics and the subdividing and merging approaches that characterize the aforementioned and further publications.

Energy Based Expansion

The basic approach for the expansion strategy is to use a combination of internal and external forces in an mass-damper-spring system for an element i:

$$m_i\ddot{\mathbf{x}}_i + r_i\dot{\mathbf{x}}_i + \mathbf{g}_i = \mathbf{f}_i.$$

Here, m_i represents the mass, r_i a damping coefficient, $\mathbf{g}_i$ the sum of internal and $\mathbf{f}_i$ the sum of external forces. $\mathbf{x}$ and its first and second derivatives $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ are the position, velocity and acceleration of the element i respectively.

Internal Forces As internal forces, a variety of authors, e.g. [30–33], use a combination of two forces: The spring force and a bending force. The spring force can be interpreted as a force that spreads a deformation of one vertex $\mathbf{x}_i$ onto its neighbourhood and can be expressed with

$$f_{s,i} = \sum_{j \in \mathcal{N}(\mathbf{x}_i)} \left(c_{ij} \frac{||\mathbf{x}_j - \mathbf{x}_i|| - l_{ij}}{||\mathbf{x}_j - \mathbf{x}_i||} \mathbf{x}_j - \mathbf{x}_i \right),$$

where c_{ij} is the stiffness between node i and j, $\mathcal{N}(\mathbf{x}_i)$ represents the set of neighbouring nodes of i and l_{ij} the natural length of the spring between node i and node j.
The second internal force, the bending force, helps to smooth the curvature of the surface represented by the triangulated mesh. It is calculated as

$$f_{b,i} = c(\mathbf{x}_i) - \mathbf{x}_i - \frac{1}{n(\mathbf{x}_i)} \sum_{j \in \mathcal{N}(\mathbf{x}_i)} (c(\mathbf{x}_{ij}) - \mathbf{x}_{ij})$$

with $n(\mathbf{x}_i)$ being the number of neighbours, $\mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ and $c(\mathbf{x}_i)$ the barycenter of the neighbours of node i that can be calculated as

$$c(\mathbf{x}_i) = \frac{1}{n(\mathbf{x}_i)} \sum_{j \in \mathcal{N}(\mathbf{x}_i)} \mathbf{x}_{ij}.$$

Both forces are summed up and weighted by user defined parameters w_s and w_b and give the internal force g_i for node i

$$g_i = w_s f_{s_i} + w_b f_{b,i}.$$

External Forces A common approach for computing external forces is to use two different forces:

- inflation force
- edge force.

Here, the inflation force is implemented in order to assure the growth/shrinking of the seed model until a possible boundary of the object is reached. Therefore, this force is directed in direction of the normal of a vertex or triangle and simply pushes the mesh until a voxel is reached in which the boundary could lie.
Chen and Medioni [33] determine the maximal possible spring force and adapt the scalar value k of the inflation force to be greater.
Park et al. [32] suggest to apply two user defined lower and upper thresholds T_{low} and T_{high}:

$$k = \begin{cases} +1 & T_{low} \leq I(\mathbf{x}_i) \leq T_{high} \\ -1 & \text{else.} \end{cases}$$

Lachaud and Montanvert compute a continuous scalar field $\Pi(\mathbf{x})$ of the volumetric image by tri-linear interpolation and calculate the inflation force by evaluating

$$k = \alpha \left(\Pi(\mathbf{x}_d) - \Pi(\mathbf{x}_i) \right)$$

with $\Pi(\mathbf{x}_d)$ being the value of a desired iso-potential surface and α a user specified weighting factor.
For the edge force the gradient of the volumetric image is used for pushing the mesh towards local minima or maxima. For computing the gradient, Lachaud and Montanvert [34] use the Sobel operator and produce a continuous vector field by tri-linear interpolation. Park et al. [32] use the 3D Monga-Deriche edge-detector [35] to produce a intensity edge field and afterwards tri-linear interpolate using the eight surrounding voxels to compute the gradient field.

Dynamical Topology

In order not to rely on the number of vertices and triangles of the seed model, different approaches for splitting triangles and merging them have been proposed.

Merging Triangles Lachaud and Montanvert [34] introduced the concept of merging two parts of a mesh if their distance is below a certain threshold. Therefore, they first create intermediate points between those parts and then triangulate between those two parts. Chen and Medioni [33] search for long and thin triangles and switch the shared edge in order to create two better posed triangles. Fig. 1.1 illustrates this issue.

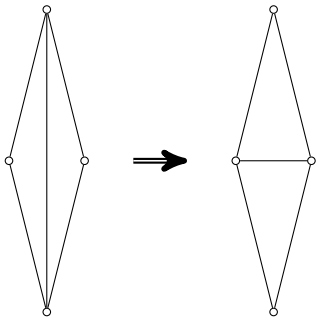

Figure 1.1: Inversion of the shared edge to create better posed triangles

Park et al. [32] search for edges below a certain threshold and merge their vertices as illustrated in fig. 1.2.

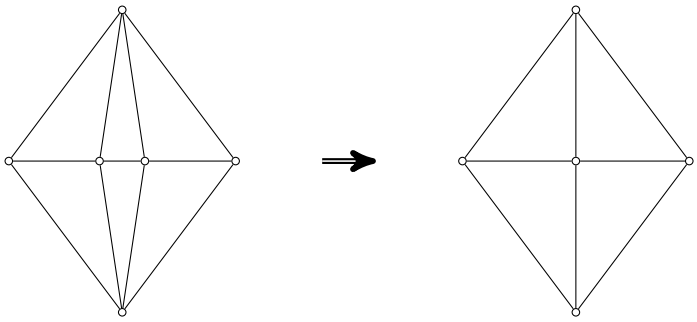

Figure 1.2: Merging two vertices whose distance is below a threshold

Splitting Triangles The issue of creating new triangles is more complex than merging. The first method for subdividing a triangle was introduced by Vasilescu and Terzopoulos [30]. The basic concept is to cut all three edges and connect the new constructed vertices as illustrated in fig. 1.3.

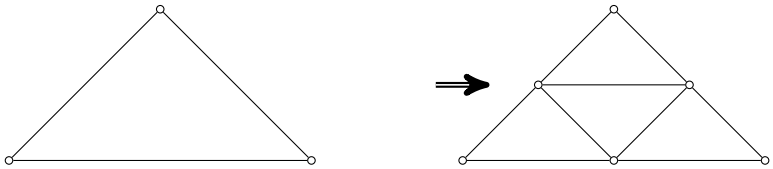

Figure 1.3: Splitting a triangle into four smaller triangles

Chen and Medioni [33] implemented a different method. After finding the longest edge in a triangle, this edge is split into two edges. The new vertex is connected to the to this edge opposing vertex and consequently, two new triangles are constructed. As the split edge is shared with another triangle, this triangle has to be split as well. Chen and Medioni therefore search in this triangle for the longest edge and split it as well. Successively, this cut has to be propagated to the next triangle. For an illustration, see fig. 1.4. Here, the upper triangle is split and propagated to the underlying triangle. This is as well split and further propagated to the lower left triangle.

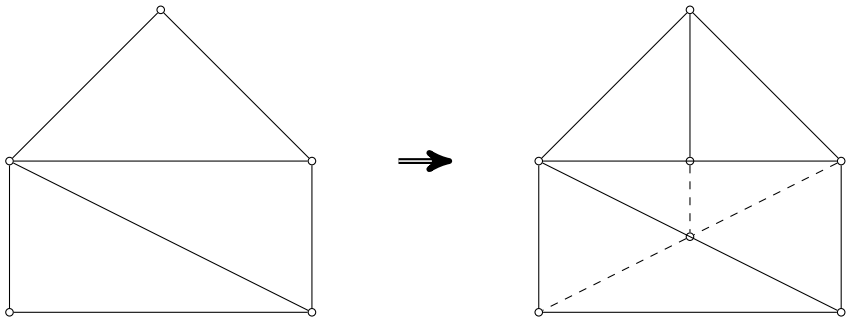

Figure 1.4: Splitting a triangle and propagating it onto the following. The dashed lines represent the propagated cuts.

Lachaud and Montanvert [34] have a similar but simpler approach. Instead of propagating the cut up to the next triangles longest edge, they simply cut the shared edge and connect it in both triangles to the opposing vertex. As an illustration serves fig. 1.5.

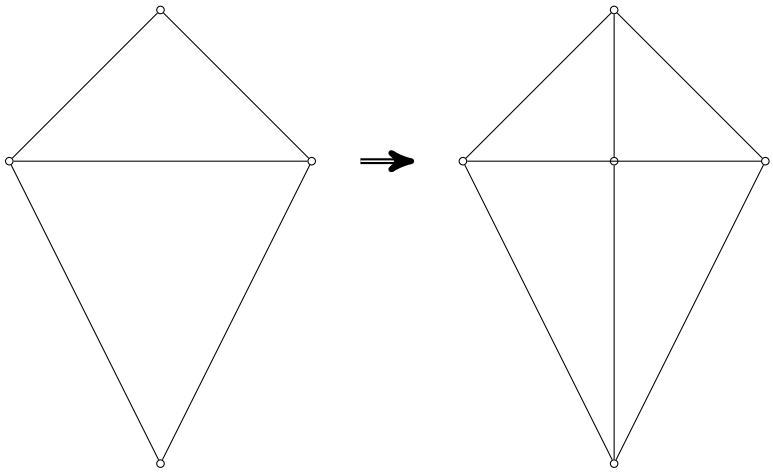

Figure 1.5: Splitting a triangle into two smaller triangles

Chapter 2

Methods

This chapter introduces a general concept of the scanning algorithm. A key method for most of our applications is the so called voxelspace which we briefly describe in the next section. In fig. 2.1 an overview of the whole scanning algorithm is illustrated. First the scanning stage is performed. In this stage, each scan is taken and has to be planned, therefore, the next best viewpoint has to be determined. This stage is succeeded by the preprocessing step. Here, the scans are registered and filtered. As final step, the surface reconstruction is performed with its two stages: inflating and detailing.
After introducing the voxelspace, we will look on the scanning process which is followed by the preprocessing phase. After that we introduce our surface reconstruction algorithm.

2.1 Volumetric Discretisation

The voxelspace is a commonly used method to efficiently describe volumetric information. Its concept is to divide a certain space into many small cubes in order to discretise that space. Each of those cubes represents a small fraction of the given space. Further information, such as being occupied or colour can be assigned to each voxel.
An efficient implementation of a voxelspace is the so called octree. Here, the space is combined in one big cube. This cube is then split into eight smaller cubes, that again can be split into eight further cubes. This procedure is done until a given minimum edge length is reached. The lastly produced cubes then represent the voxels. Voxels or cubes having the same state can be merged into their parent cube which leads to an efficient way of storing volumetric information. Furthermore, the tree structure of an octree leads to a fast access rate on the single voxels. An illustration for an octree can be found in fig. 2.2.
One application for a voxelspace is the creation of maps of the environment for collision avoidance and path planning [36, 37]. A further application is to use a voxelspace for scanning and building a volumetric representation of rigid objects.

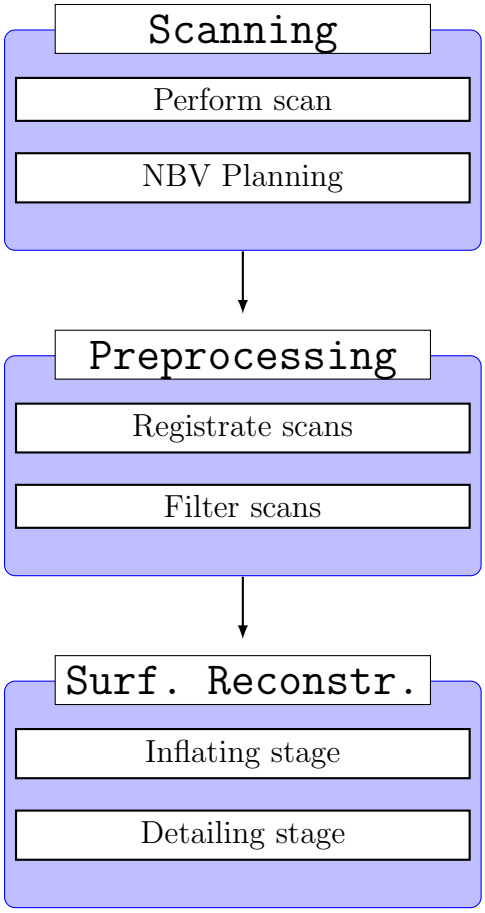

Figure 2.1: Overview of the scanning and surface reconstruction algorithm

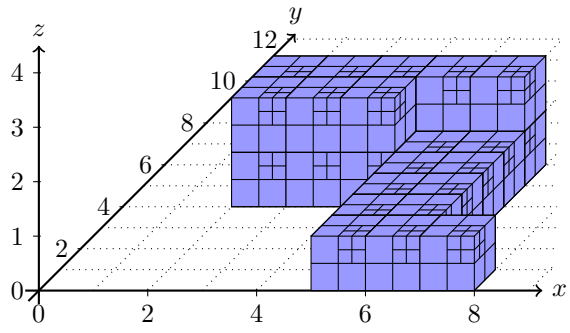

Figure 2.2: Example for a voxelpace representation using octrees

In both cases through measuring free space, the model or map is carved out of a given space with unknown state. Additionally, surface information can be inserted into the voxelspace by setting the state to a value representing occupied volumes.

2.2 Scanning

In order to acquire the necessary measurements, viewpoints need to be generated, meaning poses from where and in which direction the next measurement should be performed. Therefore, we look onto the key points of our next best view (NBV) planning algorithm in the following subsection.

2.2.1 Next Best View Planning

The NBV stage consists of two steps: First possible next viewpoints are generated which are in the second step evaluated with respect to their simulated information gain.
As our surface reconstruction algorithm differs in various points to the used method in [6,7] we have different requirements for possible viewpoints. For the filtering stage we need more than one measurement for each small part of the surface in order to increase estimation confidence. Consequently, each voxel needs a higher number of depth points attached to it. This leads to scanning the same part of the to be scanned object more than once.
For generating viewpoints, a voxelspace is generated in which five possible states are present:

Unknown voxels for which no information was gained yet

Free voxels between the camera position and the measured surface

Occupied voxels in which a surface was measured

Possible inside voxels that are behind the measured surface

Possible border voxels that are between *unknown* and *possible inside* voxels

For each new scan, this voxelspace is then updated and the given states with exception of the *occupied* state, can be freely changed to one another. From this space possible viewpoint candidates are then generated. Therefore, we form the set $\mathcal{B}$ with magnitude n that consists of all *occupied* and *possible border* voxels. For each center of those voxels b_i, $i \in 1, \ldots, n$ we search for voxels with the state *possible inside* in a certain neighbourhood and get set $\mathcal{I}_i$ of voxels with magnitude m. Using the connecting vectors $d_{j,i}$ that connect the center of a voxel l_j in $\mathbb{I}_i$ with $j \in 1, \ldots, m$ of $\mathbb{I}_i$ to b_i, the following formulas gives the viewpoint p_i and the view direction v_i with a manually determined distance d_c:

$$\bar{d}_{j,i} = \frac{\sum_1^m d_{j,i}}{\sum_1^m |d_{j,i}|}$$
$$p_i = d_c \cdot \bar{d}_{j,i} + b_i$$
$$v_i = -\bar{d}_{j,i}$$

Using this generation scheme, a massive amount of viewpoint candidates are generated which we first filter by searching for similar viewpoints that were already used. As second, we search for similar viewpoints in the set of generated viewpoints. Therefore, we look in a spherical neighbourhood for each of the points and calculate the angle α between the view directions v_i and v_j using

$$\alpha = \arccos\left(\frac{v_i \circ v_j}{|v_i| \cdot |v_j|}\right)$$

where $\circ$ denotes the scalar product. By determining similarity using a threshold, we can combine several viewpoints by averaging point and direction of all neighboured candidates.
In order to decide which of those candidates to use, we first need to evaluate their possible information gain. For that, we simulate a scan by calculating their measurements with a given depth. If a beam, meaning the connection from viewpoint to simulated measurement, penetrates an occupied voxel, this beam is adapted to end in this voxel. Else this beam ends at the given maximum beam length. Obviously, this method does not accurately calculate the real information gain but produces a good estimate given the already scanned data. By counting all penetrated voxels weighted differently for their states, we gain an indicator q_i for this scan's information gain:

$$q_i = w_{occ} \sum^{N} v_{occ} + w_{ins} \sum^{M} v_{ins} + w_{unk} \sum^{O} v_{unk}$$

where N is the number of *occupied* voxels v_{occ}, M of the *possible border* and *possible inside* voxels and O is the number of *unknown* voxels. Furthermore, w_{occ}, w_{ins} and w_{unk} are factors for weighting the different states of the voxels. For having overlapping scans w_{occ} should be the largest factor. By adapting w_{ins} the user can determine the exploration affinity of the next best scan. As most of the voxels have the state unknown in an early stage of the scanning process, v_{unk} should be significantly smaller than the other two factors. In our experiments we determined $w_{occ} = 1$, $w_{ins} = 0.5$ and $w_{unk} = 0.01$ to be a suitable choice. For the occupied voxels, we only use those where the number of actual scanned depth points inside this voxel is below a certain threshold. w_{unk} usually does not greatly affect the evaluation of the viewpoint candidates. We chose this parameter to be a fail safe measurement that only influences the case that no other voxels than unknown can be seen and helps for exploration.
Afterwards, the scan corresponding to the greatest q_i is picked as the next best view.

2.3 Preprocessing

In this stage, we try to prepare the scanned data in order to fit to our surface reconstruction algorithm.

2.3.1 Registration

As the given hardware lacks of precise pose measurement, we use a given implementation of the ICP algorithm. For the correspondence search, this implementation searches for the closest points in a given radius. Points are rejected on base of the similarity of their normals.
The registration is implemented in two stages: In the first stage, we directly try to find the transformation that aligns the current measurement to measurements taken and registered before. If a given number of correspondences is found then this measurement is aligned and stored. If there are too few correspondences, the current measurement is stored for later processing.
In the second stage, all measurements where too few correspondences were found are processed again until all measurements are aligned or a given number of iterations is reached.
As results in sec.3.2 imply, for the given data, the registration achieves an significant improvement but pose and measuring errors remain. Therefore, we introduce in the following subsection a filtering step, which helps to cope with the noisiness of both the measurement and its pose.

2.3.2 Filtering

After performing several scans, we need to extract the necessary information and to reduce redundancy of the gained data. Therefore, we applied a filtering stage in our algorithm which consists of a normal estimation step and a combining and reducing step which are applied on the whole data set.
For estimating the normals for each depth point p_i, we search the nearest two depth points $p_{n,1}$ and $p_{n,2}$. By using the cross product denoted by $(\cdot) \times (\cdot)$ we gain the normal n_i:

$$n_i = \overline{p_i - p_{n,1}} \times \overline{p_i - p_{n,2}}$$

with $\overline{(\cdot)}$ for denoting the normalized vector. As the angle β between the normal and the negative view direction $-d_{v,i}$ can not be greater or equal to 90, we can perform a sanity check by calculating this angle:

$$\beta = \arccos(\overline{n_i} \circ -\overline{d_{v,i}}).$$

In the case that β is near to 90, we search for the next nearest point and calculate again the normal, until the sanity check is fulfilled. Another possibility is that the angle β is pointing in the wrong direction. In this case the actual normal is calculated by multiplying n_i with (-1).

In order to diminish noise and reduce the amount of measurements, we now search for each point in a given reduction radius r_{red} for nearby points and denote this set of points as $\mathcal{R}$. For each of them, we search in the radius of estimated noise r_{noise} and only use those points that are located in a small area around the direction of the normal and negative normal for nearby points. By comparing their normals, only similar points are used for calculating the expectation value of this measurement. By using the measurements with the highest number of similar measurements and neglecting all others of $\mathcal{R}$ we can both, reduce point density and noise.

2.4 Surface Reconstruction

The implemented surface reconstruction algorithm consists of two different stages and two measures to allow for a smooth mesh with regularized triangles. First we look on the inflating stage which is applied on the filtered data. After that, we explain the concept of the detailing stage which is then followed by the smoothing and regularizing methods. Fig. 2.3 gives an overview over the surface reconstruction algorithm.

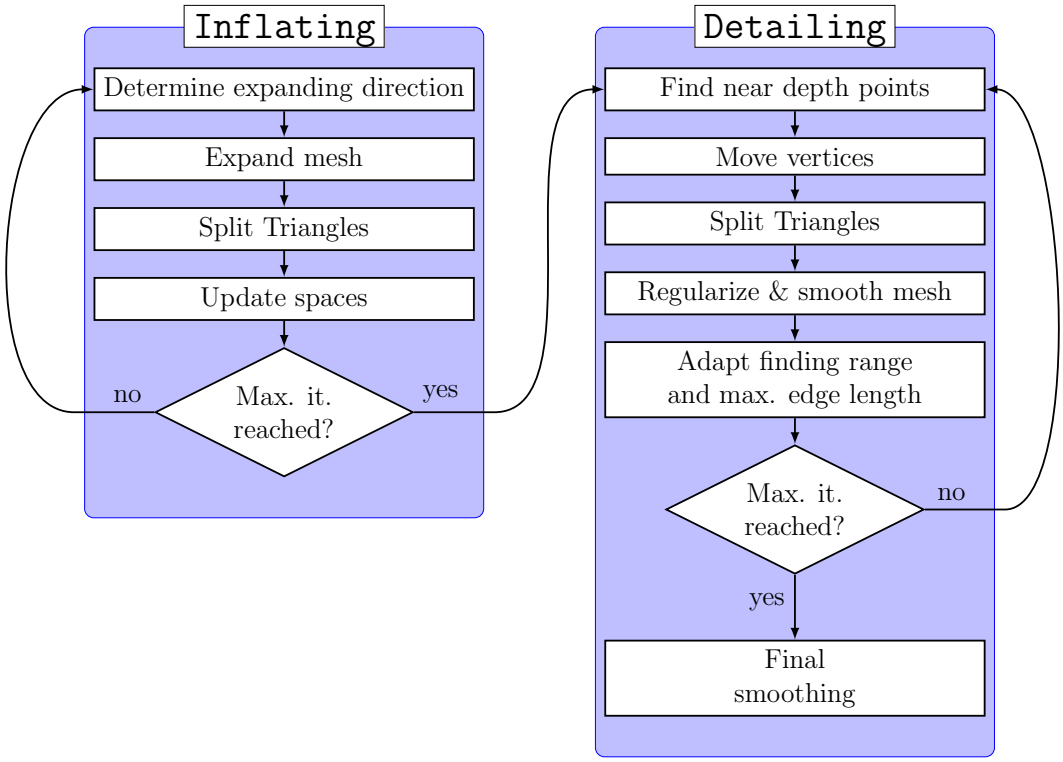

Figure 2.3: Overview of the surface reconstruction algorithm

2.4.1 Inflating Stage

The inflating stage aims to build a raw model of the scanned object. For that we need two different voxelspaces:

Surface Space This space represents the surface of the scanned object. It is initialized by setting the state of those voxels occupied that have located at least one of the remaining depth points inside them. All other voxels are set to the state free.

Inside Space This space is used to represent the volume inside the mesh model. It is initialized as free. Each of its voxels that is intersected by a triangle of the mesh model is set to occupied. If a triangle is moved and does not longer intersect with a certain voxel, this voxel remains occupied but is now noted as being inside the mesh model. This is verified by ensuring that the mesh model can only expand.

The inflating stage is initialized with a small seed model - in our case a tetrahedron is sufficient - that fits completely in one voxel of the inside space. It is then expanded until at least one voxel fits inside the seed model and is no longer intersected by one of the seed model's triangles. Consequently, we have at least one "inside" voxel.
This issue is illustrated in fig. 2.4, where the blue square represents the one necessary "inside" voxel.

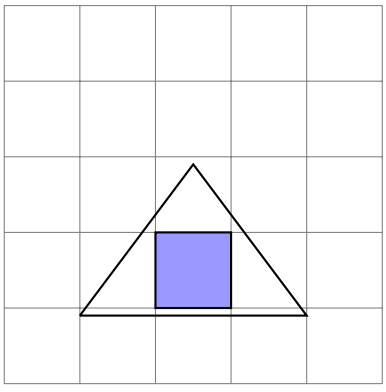

Figure 2.4: Through cut of the seed model in a voxelspace

In the next step the actual inflating stage begins. This stage is a iterative stage in which several steps are processed sequentially.
Fig. 2.5 gives a simple example of a mesh model in the inflating stage. The grid represents the voxelspace but without any states assigned. We will use this example for illustrating the following steps.

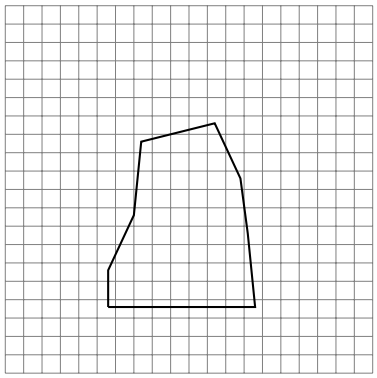

Figure 2.5: Through cut of a complex 3D model

The occupied voxels of the two spaces in this example are illustrated in fig. 2.6 as red filled rectangles for the surface space and blue cross hatched rectangles for the inside space.

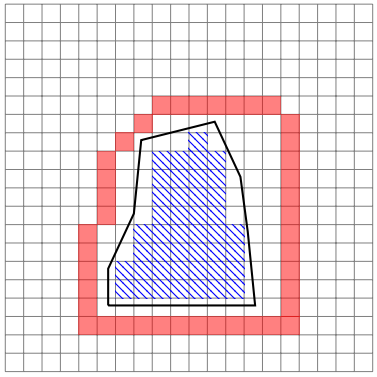

Figure 2.6: Through cut of a complex 3D model with surface space as red filled and inside space as blue cross hatched rectangles

Determine expanding direction

To determine for each vertex of the mesh model the direction it has to expand into, we search in the inside space for the set M of voxels that are within a certain distance to this vertex. Combining the connecting vectors c_i from the center of those voxels to the vertex and normalizing them gives the direction vector d_i:

$$d_i = \sum^{M} \frac{c_i}{|c_i|}.$$

Using the latter example, fig. 2.7 illustrates the procedure for determining the expanding direction. Here, the vertices of the mesh are represented by red circles. The voxels of the inside space that are within a defined distance to the lower left vertex are additionally cross hatched with red lines.

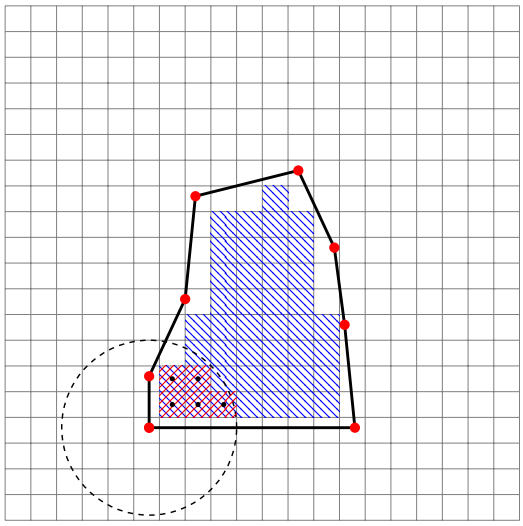

Figure 2.7: Through cut of a complex 3D model with red circles as vertices. Inside space used for expanding direction is additionally cross hatched with red lines

Now we have to verify that the following movement of the single vertex is not directed into the inside of the mesh model. This is done by initializing a beam that starts at the vertex and its direction is d_i. If this beam intersects with another triangle then it is pointed to the inside and cannot be used for expansion. If otherwise, a second check has to be performed in order to prevent the model from expanding through a "hole" in the surface space. Therefore, we use the same beam as for the first check and search in all intersected voxel of the surface space for a occupied one. If there is a intersection then this vertex can safely be moved in direction of d_i.

Vertex Movement and Space Update

In the next step all vertices with a safe direction vector are moved depending on their distance $d_{s,i}$ to the next occupied voxel of the surface space. Therefore, the moving distance $d_{m,i}$ is calculated by

$$d_{m_i} = k \cdot d_{k,i}$$

with $k < 1$ where the convergence rate to the surface can be adjusted. A movement greater than the edge length l of a voxel is not allowed as this would produce holes in the inside space. Therefore, $d_{k,i}$ is limited to the edge length and calculated as

$$d_{k,i} = \begin{cases} d_{s,i} & , d_{s,i} \leq l \\ l & , \text{else.} \end{cases}$$

After moving all vertices, the triangles with an edge length greater than $2l$ are split and the inside space is updated. Then this procedure is repeated until all voxels that can safely be moved are located within a distance of $d_{stop} = 0.5l$.

Fig. 2.8 portrays the update vectors with a length of d_{stop} for each vertex of our example with a black arrow. If an update vector does not reach the surface space, represented by the red filled rectangles, the corresponding vertex can safely be moved and is coloured in green. In this case we set one of the occupied voxels of the surface space to be free in order to illustrate a hole in this space. All vertices that can not be moved, either because they are to close to the surface or because there is a hole in direction of the update vector, are coloured in red.

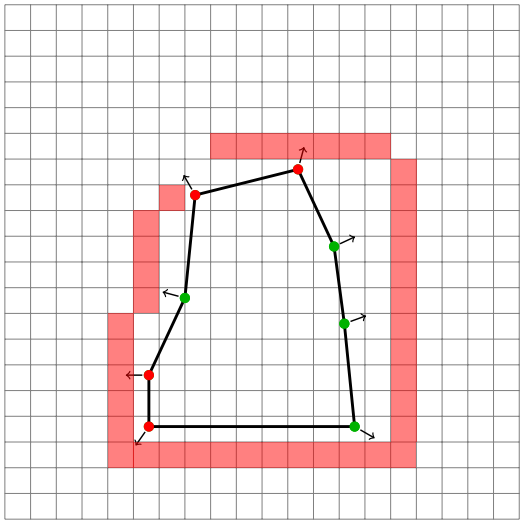

Figure 2.8: Through cut of a complex 3D model with circles as vertices. Vertices that can safely are green, else red.

2.4.2 Detailing

In the detailing stage the raw model is refined in order to recover details. Therefore, we utilize an iterative procedure that searches for nearby depth points and move the vertices of the 3D model towards them. This is done with an adaptive finding range l_f in order to smoothly refine the model. It starts with a range of $2l$ where l is again the edge length of a voxel in inflating stage. It is then iteratively reduced to the minimum edge length l_m of the model and is calculated as:

$$l_f = l_m + ((i_{max} - i_{cur})/i_{max} \cdot (l - l_m))$$

where i_{max} is the maximum iteration number and i_{cur} the current iteration number. Furthermore, l_f is used to adapt the edge length of the triangles. Therefore, after each iteration all edges are checked and their corresponding triangles are split according to the method of Lachaud and Montanvert [34] if their length is greater than l_f.

2.4.3 Smoothing and Regularizing

Inflating a mesh model leads to irregular triangles and rough surfaces. A key point to a good mesh model is therefore smoothing and regularizing technologies. In the following subsection we explain two methods that can be applied in order to cover those.

Smoothing

In order to smooth sharp edges that occur due to the surface reconstruction algorithm, local surface smoothing needs to be applied on the build mesh. We concentrate on a method which uses the implemented triangle topology with its information about the local triangle neighbourhood. For that first normals for each vertex have to be estimated. The normal $n_{t,i}$ for triangle t_i can be produced by calculating the cross product denoted by $(\cdot) \times (\cdot)$ of two of its edges e_1 and e_2 by using

$$n_{t,i} = \frac{e_1}{|e_1|} \times \frac{e_2}{|e_2|}.$$

Attention has to be paid to the correct order for e_1 and e_2 to produce a normal that points outside the mesh model. By averaging the normals of all m triangles a certain vertex v_i is part of we can estimate the normal $n_{v,i}$ for this vertex

$$n_{v,i} = \frac{\sum_{i=1}^{m} n_{t,i}}{\sum_{i=1}^{m} |n_{t,i}|}.$$

For the next step we need to estimate the local change of the surface. Therefore, we collect for the triangle t_i from the three neighbouring triangles those vertices that are not shared with t_i and denote this set as $\mathcal{N}_i$. For each vertex v_j in $\mathcal{N}_i$ we can now calculate the distace $d_{j,i}$ between vertex v_j and a virtual surface represented by the normal $n_{v,i}$ with

$$d_{j,i} = n_{v,i} \circ (v_j - v_i).$$

In fig. 2.9 this issue is illustrated. The blue dashed line represent the virtual surface of v_i and the distance $d_{j,i}$ is drawn as doubled arrow. Please note, that the surface of the normal $n_{v,i}$ does not correspond to the surface represented by the triangle t_i. This is due to the fact that each vertex is part of more than one triangle and therefore using only one triangle's surface would not fully represent the local surface.

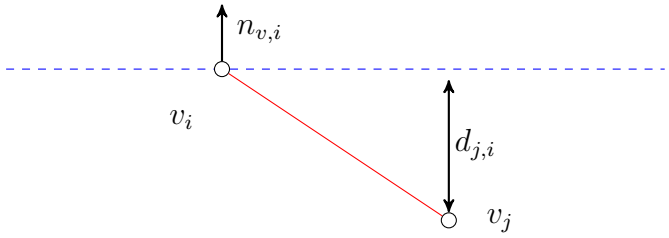

Figure 2.9: Illustration of vertex v_j and its counter vertex $v_{c,j}$

The angle $\alpha_{i,j}$ between the surfaces of triangle t_i and its neighbour t_j can be calculated with

$$\alpha_{i,j} = \arccos\left(\frac{n_{v,i} \circ n_{v,j}}{|n_{v,i}||n_{v,j}|}\right).$$

As next step we map each vertex of $\mathcal{N}_i$ to its counter vertex $v_{c,j}$ of the triangle t_i. Meaning, we form a vertex pair with v_j and that vertex of t_i that is not common in both triangles. An illustration can be found in fig. 2.10. Here, the red and blue filled circles represent v_j and $v_{c,j}$, respectively.

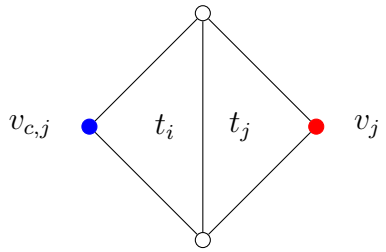

Figure 2.10: Illustration of vertex v_j and its counter vertex $v_{c,j}$

By making use of $d_{j,i}$ and $\alpha_{i,j}$ we can now smoothen the surface in a local area. Therefore, we move v_j towards the virtual surface and vice versa, the surface towards v_j, meaning that v_i is moved in the opposite direction. For that the update value u is calculated as

$$u = k_s d_{j,i} \sin((\text{conf}(v_i) - \text{conf}(v_j))\pi + 1)$$

where $\text{conf}(\cdot)$ represents the estimation confidence of a vertex and k_s is a factor to regulate the movement that has to be smaller than 0.5. In order to limit the smoothing to a certain angle range, we can adapt u with a minimum α_{min} and maximum angle α_{max}

$$u = \begin{cases} 0 & , \alpha_{i,j} - \alpha_{min} < 0 \\ 0 & , \alpha_{max} - \alpha_{i,j} < 0 \\ u & , \text{else.} \end{cases}$$

The update rule for $v_{i,k}$ and $v_{j,k}$ where the subscript k denotes the time step is then

$$v_{i,k} = v_{i,k-1} + u n_{v,i}$$
$$v_{j,k} = v_{j,k-1} - u n_{v,i}.$$

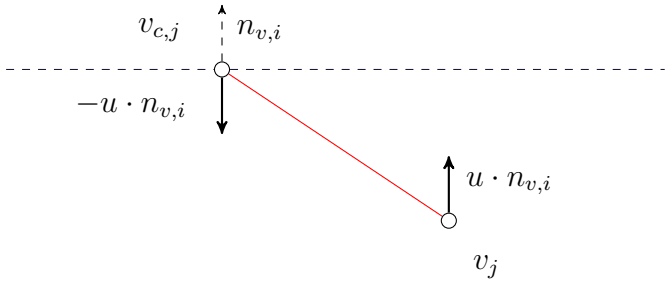

Figure 2.11: Update procedure applied on vertex v_j and its counter vertex $v_{c,j}$

Fig. 2.11 illustrates the update procedure. Here the two vertices $v_{c,j}$ and v_j are moved in direction of the normal of v_i with $-u$ and u as moving distance.

Regularizing

As only vertices are moved, spiky mesh artefacts are possible to occur. These artefacts can lead to irregularities in the mesh during the progression of the process. In order to prevent this a simple procedure helps to regularize a mesh without too much of an influence onto the model structure. An example for irregularities can be found in fig. 2.12. Here some of the triangles are long and spiky. Just by moving the vertex into the middle of the mesh, the triangles are more evenly distributed. The surface of this mesh is only slightly changed.

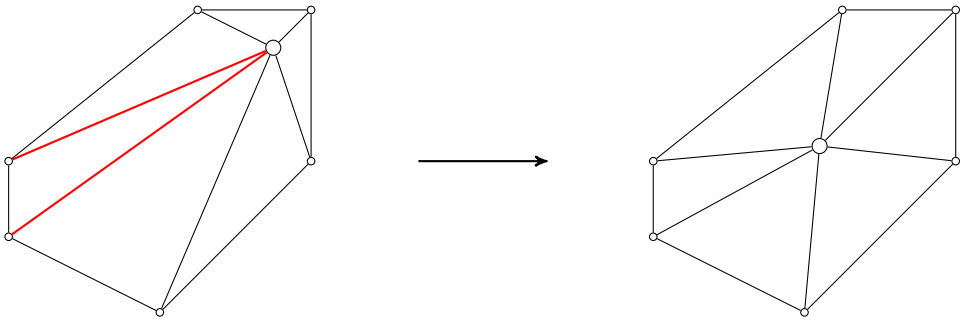

Figure 2.12: By moving the vertex in the middle the mesh can be regularized

In order to cope with such phenomenons, we need to analyse all edges of each vertex and their relation to each other. By calculating the angle between two edges of one edge, we get a hint on how regular the triangles between those edges are. For our purpose we want the three angles of all triangles to be near 60°. For example, looking onto the two red edges in fig. 2.12 we can see that their angle α_{reg} is far from being 60°. The angle between two edges e_1 and e_2 can be calculated by

$$\alpha_{reg} = \arccos\left(\frac{e_1 \circ e_2}{|e_1||e_2|}\right).$$

As we want this angle to be nearly 60° we move the cosine of α_{reg} to be almost 0 if the angle is close to 60° and denote the variable to be regulated as a:

$$a = \cos(\alpha_{reg}) - 0.5$$

In order to construct an update vector u_{vec} that is on the plane spanned through e_1 and e_2 and points orthogonal to e_2 away from e_1 we use the cross product

$$u_vec = e_2 \times (e_2 \times e_1).$$

Fig. 2.13 illustrates the construction of u_{vec}.

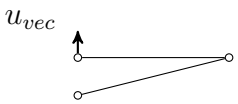

Figure 2.13: Constructed update vector u_{vec}

The update movement u_n for the not shared vertex of e_2 is then calculated with a regulating factor k_{reg} by

$$u_n = k_{reg} \cdot a \cdot u_{vec}$$

and for the shared we use the negative direction

$$u_s = -u_n.$$

By applying this procedure on all edges of a vertex, we can regularize the triangles of a mesh with only low impact on the surface.

Chapter 3

Experimental Results

In this chapter, we present results of two experiments made using real robotic system. The first experiment was performed with a KUKA omniRob platform. The interested reader is referred to [38] for more information regarding the omniRob platform. This robot is extended with a Pan Tilt Unit (PTU) on which a stereo system is mounted as illustrated in fig. 3.2. Please note that in this figure an Asus Xtion is on top of the stereo system which is not used. The stereo system consists of two Guppy Pro F-125 cameras from Allied Vision Technologies with a resolution of 1292x964 pixels, for more information refer to [39]. For gripping and holding objects, a KUKA LWR 4+ manipulator [40] is extended with a Schunk PG 70 Gripper. The whole setup is illustrated in fig. 3.1.

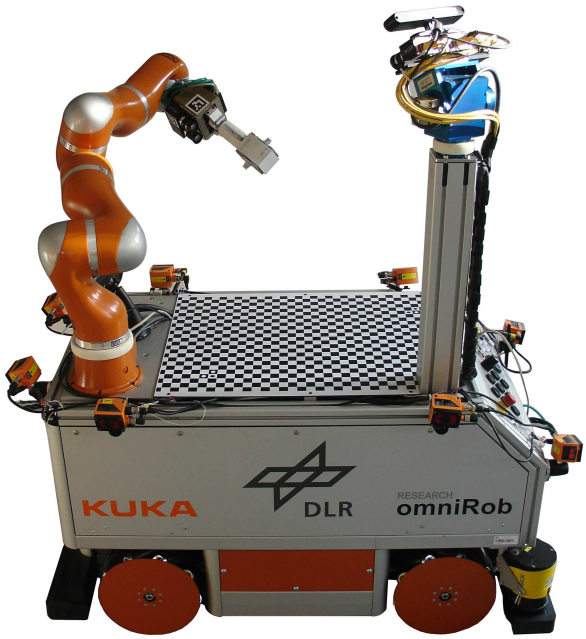

Figure 3.1: Experimental setup with KUKA omniRob and stereo system

Figure 3.2: Stereo system mounted on a PTU

For comparison reasons, we performed a second experiment using preciser hardware. The experimental setup consisted of a KUKA KR16-2 manipulator, see [41], with a ScanControl 2700 − 100 laser striper from Micro-Epsilon [42] mounted on it. Both devices are significantly more accurate in comparison to the KUKA LWR4+ and the used stereo-system. See fig. 3.3 for an illustration. For evaluating our approach we use two objects that vary in shape and texture. One to which we will refer as filter object in the following is portrayed in fig. 3.4, the other is a coffee package shown in fig. 3.5. Both objects shown in the images are exemplarily gripped with a Schunk PG 70 Gripper. Please note, that these are not the exact poses of the objects that can be seen in the following sections. All scans are compared to manually scanned meshes. These were made with a hand held high precision scanner and manually aligned. Those 3D models are shown in fig. 3.6 and fig. 3.7. In the following, we look at the different stages of our approach individually. Here we used the first experimental setup. First we will show the generation and evaluation of viewpoints which is followed by the results of the ICP registration. After the filtering phase, we examine the surface reconstruction step which is divided into the inflating and the detailing stage. Next, we compare the build models to the hand made meshes. Finally, we will examine the resulting mesh models gained with the second experimental setup and compare those models with the hand made meshes as well.

Figure 3.3: Experimental setup with KUKA KR16 and ScanControl laser striper

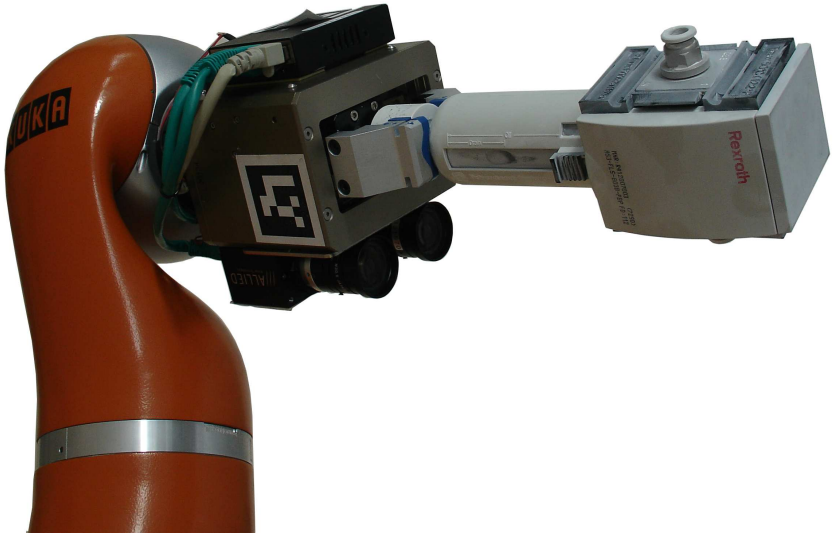

Figure 3.4: The filter object gripped with a PG 70 Gripper

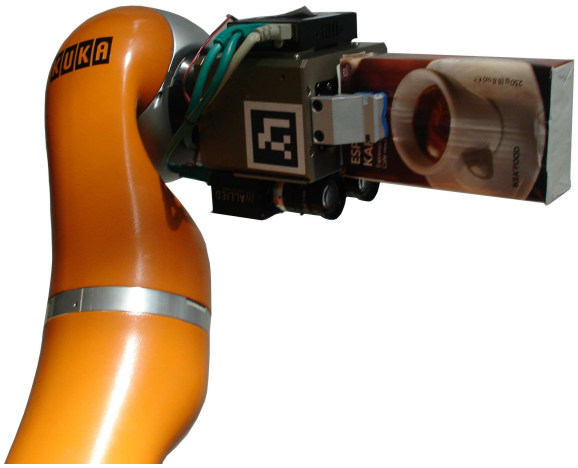

Figure 3.5: The coffee package gripped with a PG 70 Gripper

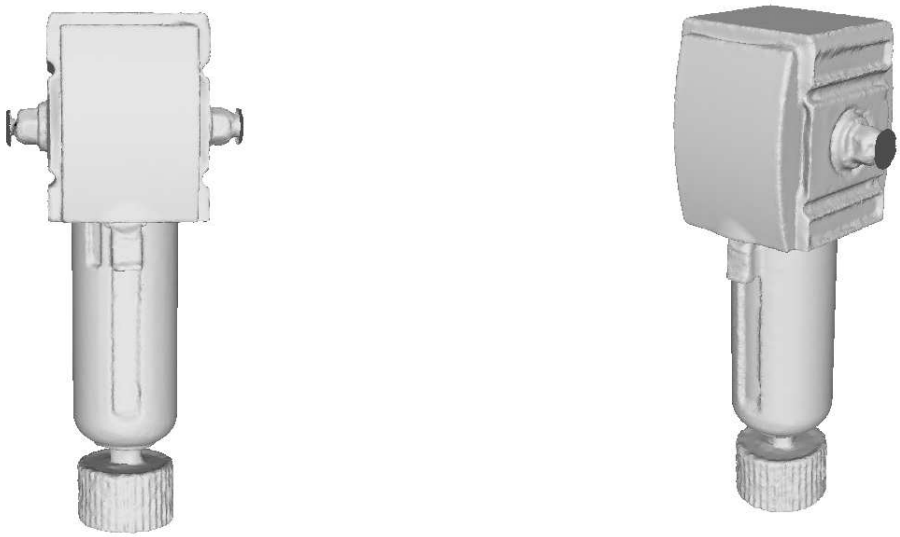

Figure 3.6: High precision scan of the filter object

Figure 3.7: High precision scan of the coffee package

3.1 Viewplanning

The first scan for measuring a new object is a predefined viewpoint that captures part of the gripper and most certainly one side of the object. Based on the first scan, the NBV voxelspace is updated. An example for this with the filter object can be seen in fig. 3.8. Here, the state of the voxels are coloured as: White for occupied voxel, grey for unknown (as initial state) and black for border voxel. Light yellow voxels represent the inside space.
Please note that as we know where the gripper is, its corresponding space is set to occupied and does not influence as well as is not influenced by the NBV planning process. Furthermore, the space around the gripper is set to unknown and is as well not influenced and does not influence this process. These two assumptions are made as the object is located above the gripper and therefore, both areas are only shown for illustration reasons.

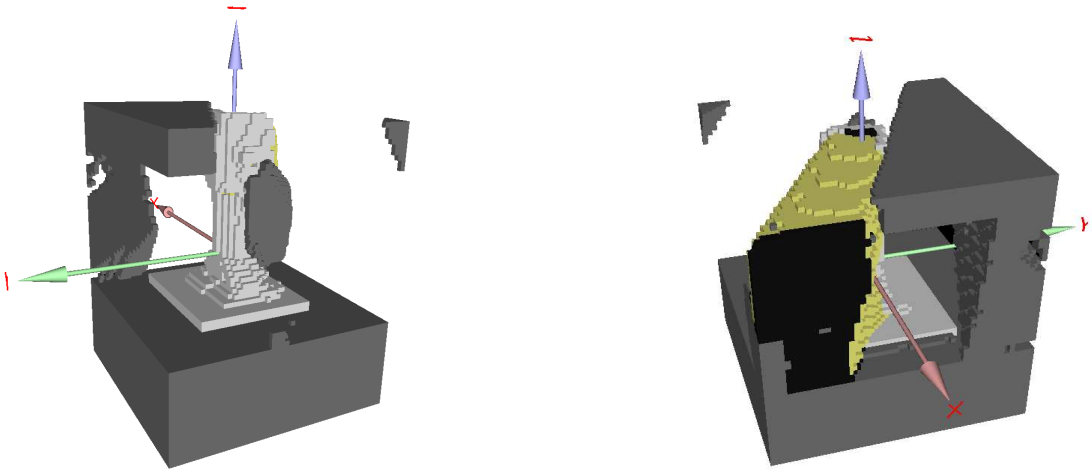

Figure 3.8: Voxelspace for determining the next best view. White voxels are occupied, grey unknown, black border voxel and yellow are inside voxel

For the following figures we will print this space for illustration reasons. The generated viewpoints can be seen in fig. 3.9. Here, the biggest part of viewpoints is located on places where both, occupied voxels and inside space most certainly can be scanned.
After filtering the viewpoints locally, the next best viewpoint is chosen. An illustration for this is given in fig. 3.10. Here, the NBV is shown as the only green dot. All other viewpoints are coloured in red.
After performing a series of scans, the final NBV space has no border voxel and no viewable inside space. In fig. 3.11 an example for the filter object is shown. Here some occupied voxels obviously can be classified as outliers. An advantage of the presented method is that those outliers do not affect the NBV planning process, as there is no inside space near to them.

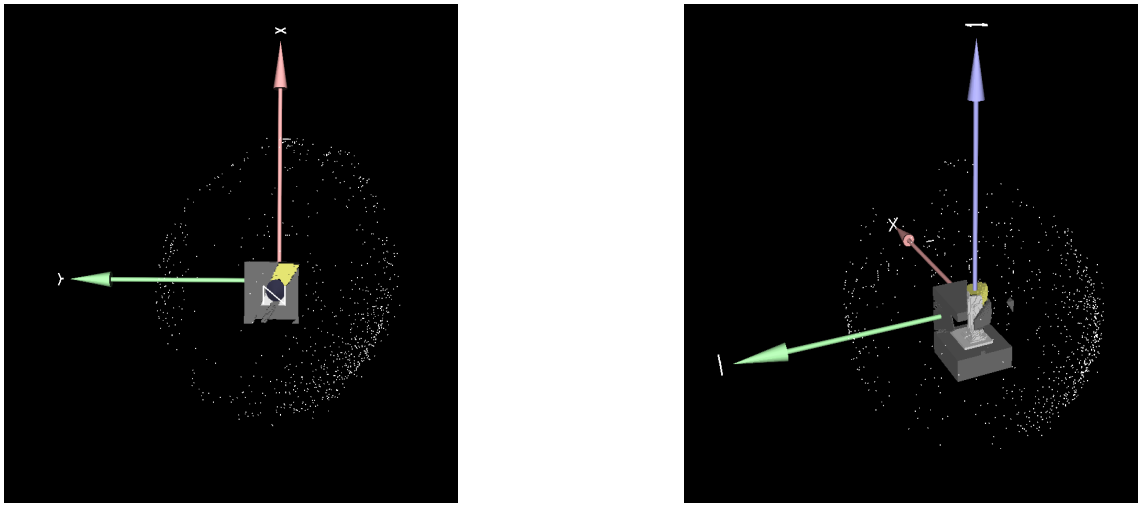

Figure 3.9: All generated viewpoints together with the NBV space

In the presented example, 33 scans where made until no viewpoints could be generated. This means, that all viewable voxels are on the one hand occupied and on the other have at least a certain amount of measurements assigned to them. In the given example, we set the minimum of scan points per voxel to 5.

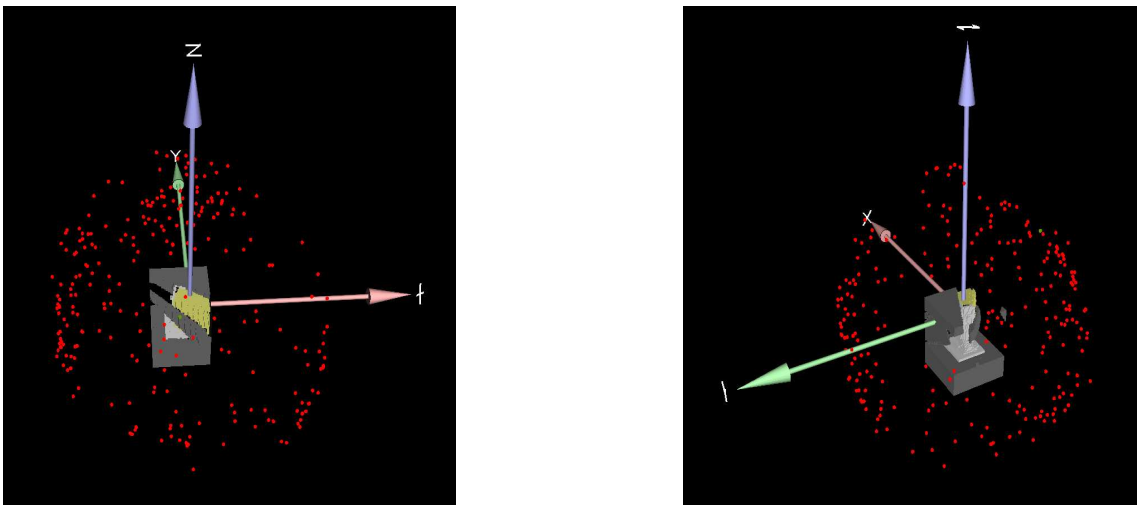

Figure 3.10: Filtered and evaluated viewpoints together with the NBV space. The green dot represents the next best viewpoint

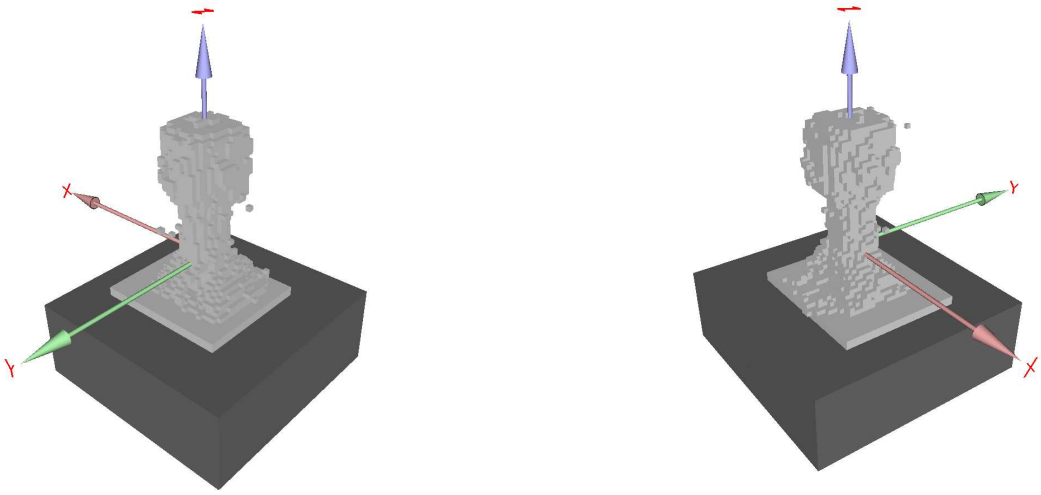

Figure 3.11: The final NBV space

3.2 Registration

After performing a series of scans, we can now start the preprocess stage of our algorithm.

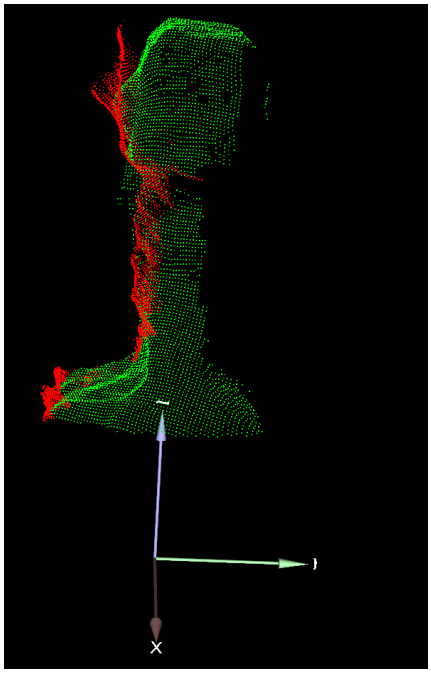

Figure 3.12: Two unaligned scans of the filter object

Fig. 3.12 and fig. 3.13 show the data of two scans - illustrated as red points for one scan and green points for the second scan - of the filter and the coffee package after applying a box filter to remove all not needed points. Obviously, those scans need to be aligned for further proceedings
After registration using the ICP algorithm a significant improvement could be achieved which can be seen in fig. 3.14 and fig. 3.15, where the white points represent the template scan. The not aligned scan is illustrated as red points and the scan after registration as green points.
Although, the alignment of both scans can be improved through ICP, an error through measuring failures can still be recognised that is too high for surface reconstruction. Fig. 3.16 illustrates this. Here, in the template image (white points) a measuring failure occurred that is highlighted with a blue ellipse. Unfortunately, this result yields for many scans taken with the given hardware. Therefore, we will take a closer look on the results of our filtering stage in the next section.

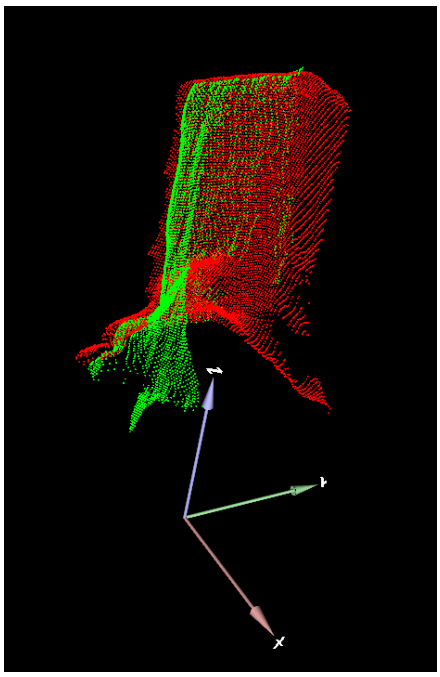

Figure 3.13: Two unaligned scans of the coffee package

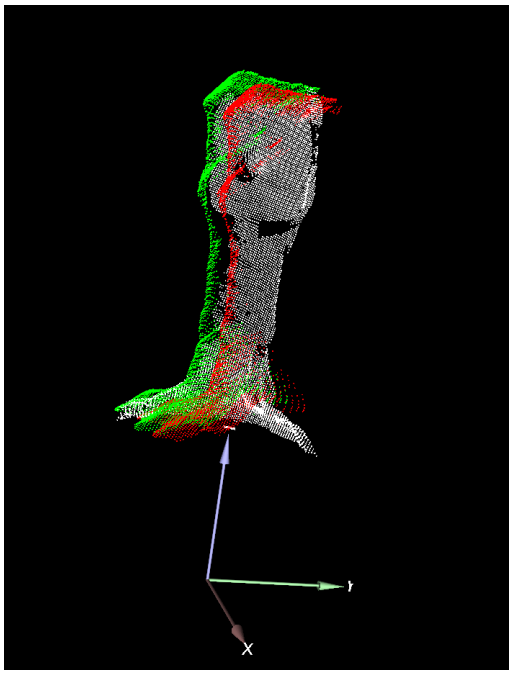

Figure 3.14: Two scans of the filter object, white points represent the template image, red points the unaligned scan and green points the aligned scan

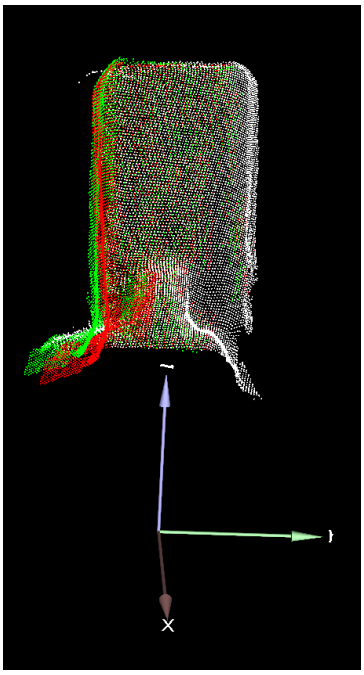

Figure 3.15: Two scans of the coffee package, white points represent the template image, red points the unaligned scan and green points the aligned scan

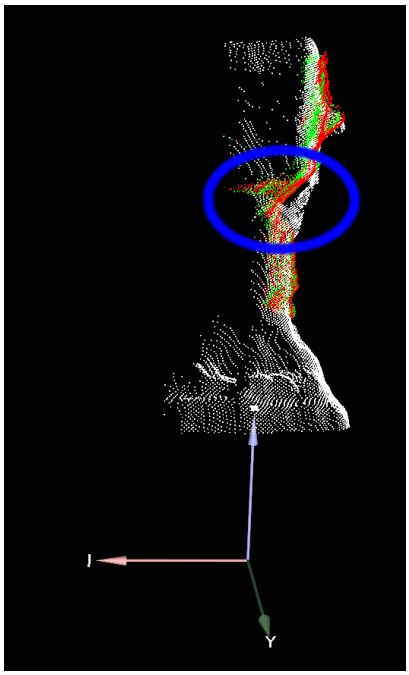

Figure 3.16: Two scans of the filter object, white points represent the template image, red points the unaligned scan and green points the aligned scan. The blue ellipse shows the remaining error through measuring failure

3.3 Filtering

As the filtering stage is applied globally on all images, fig. 3.17 and fig. 3.18 show the combined and registered scans in one image for the coffee package and the filter object.

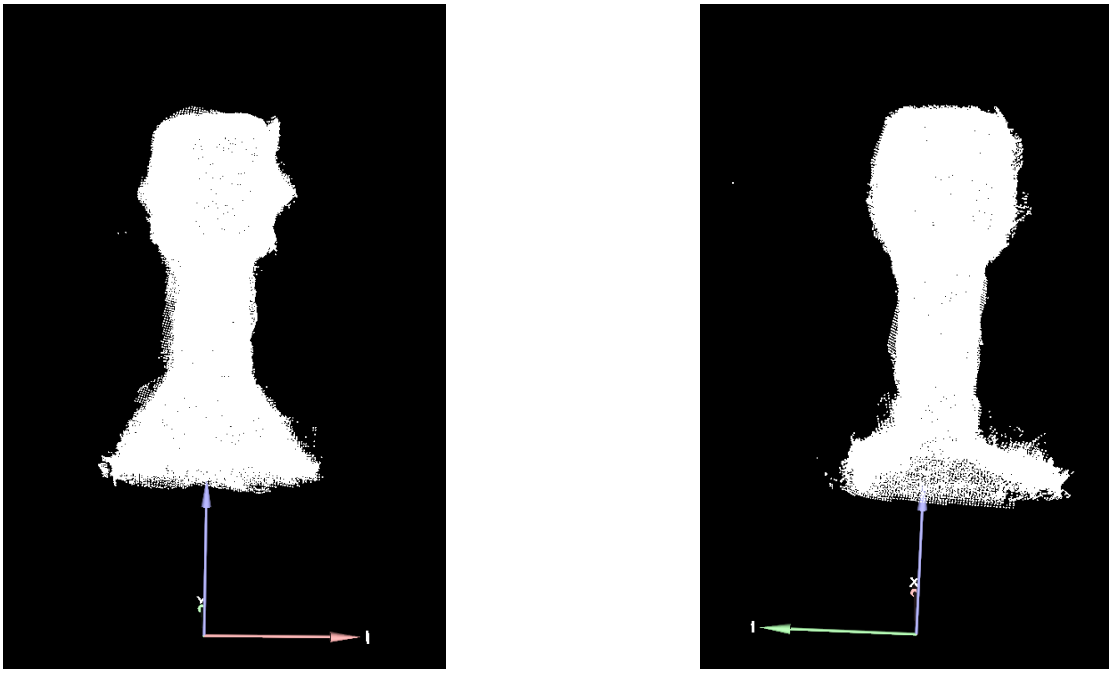

Figure 3.17: All scans of the filter object combined

After filtering those, the result is illustrated in fig. 3.19 and fig. 3.20. In both cases, the number of measurements is significantly reduced. Most of the multiple measured surfaces that are displaced are now combined to a more continuous surface. One side effect of the filtering stage is that most outliers are vanished as they have not enough neighbouring points to support the hypothesis of them being at a certain point in space.
By changing the noise radius we can trade between measurement confidence and details. The larger this parameter is, the less edges are detected, see fig. 3. Here, we highlight an edge of the real object with a blue ellipse. Obviously, this edge is compared to the original parameters blurred out.
By changing the reduction radius, the point density can be changed and outliers filtered. If this radius is too small, only the noise radius is used for determining the measurement confidence, as no direct neighbours can be found. This can lead - in case of a poorly chosen parameter - to large measurement holes, if too many points are filtered. Fig. 3.22 illustrates this example.

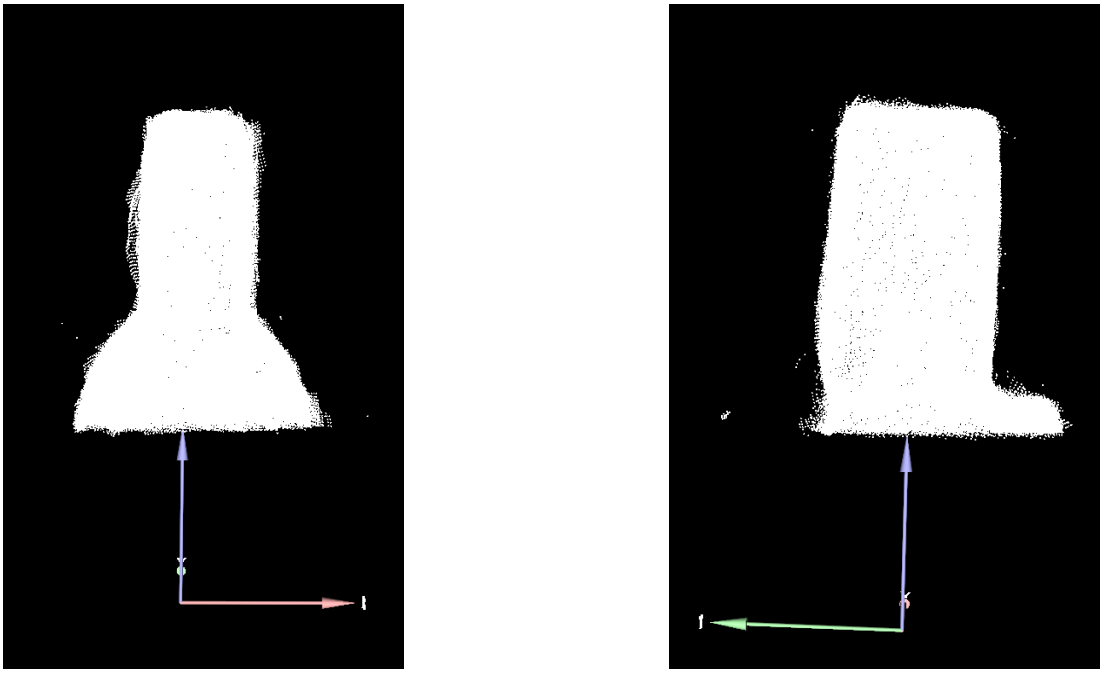

Figure 3.18: All scans of the coffee package combined

On the other hand, if the reduction radius is too large, the filter would produce a too great distance between the remaining measurements, refer to fig. 3.23. This will lead to holes in the surface space for the inflating stage and can be seen in fig. 3.24. Another side effect of this is that again, small details would be lost by averaging the measurements.

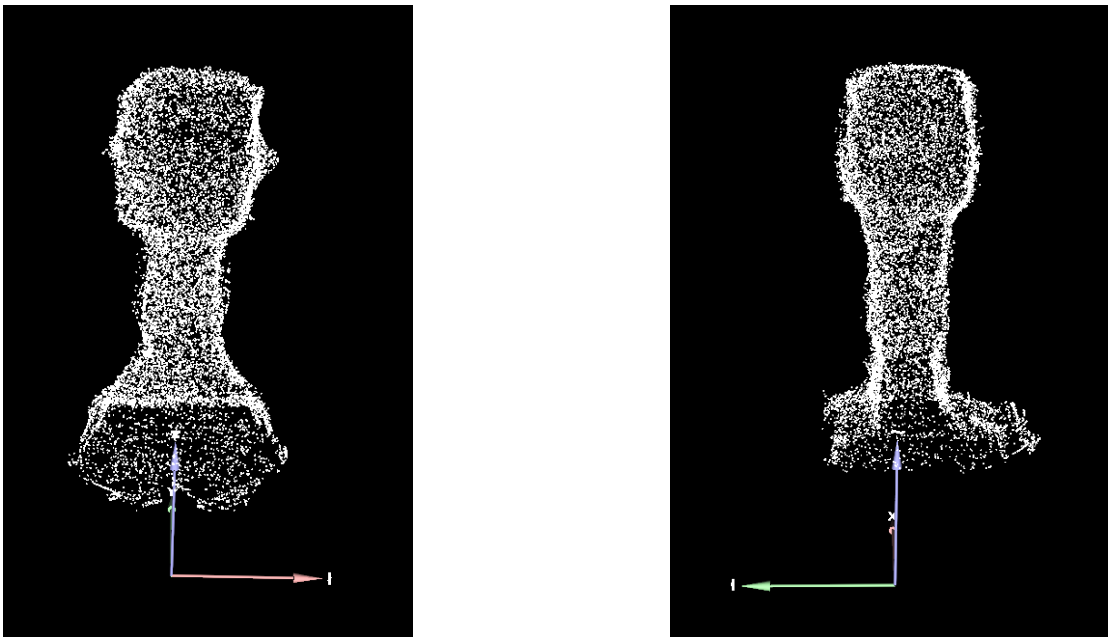

Figure 3.19: All scans of the filter object combined after filtering

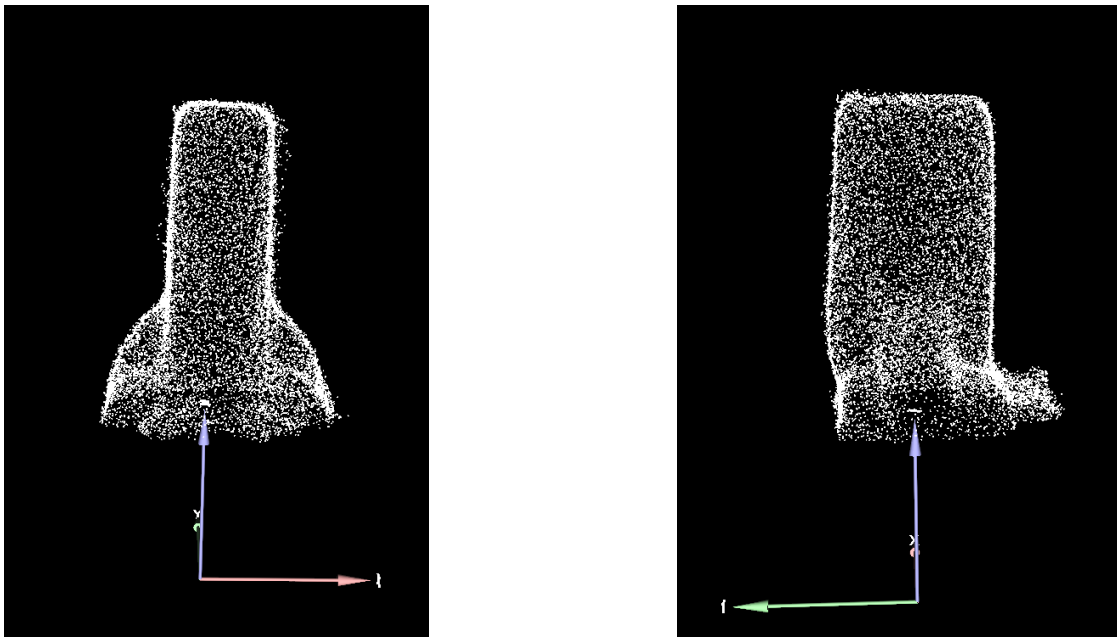

Figure 3.20: All scans of the coffee package combined after filtering

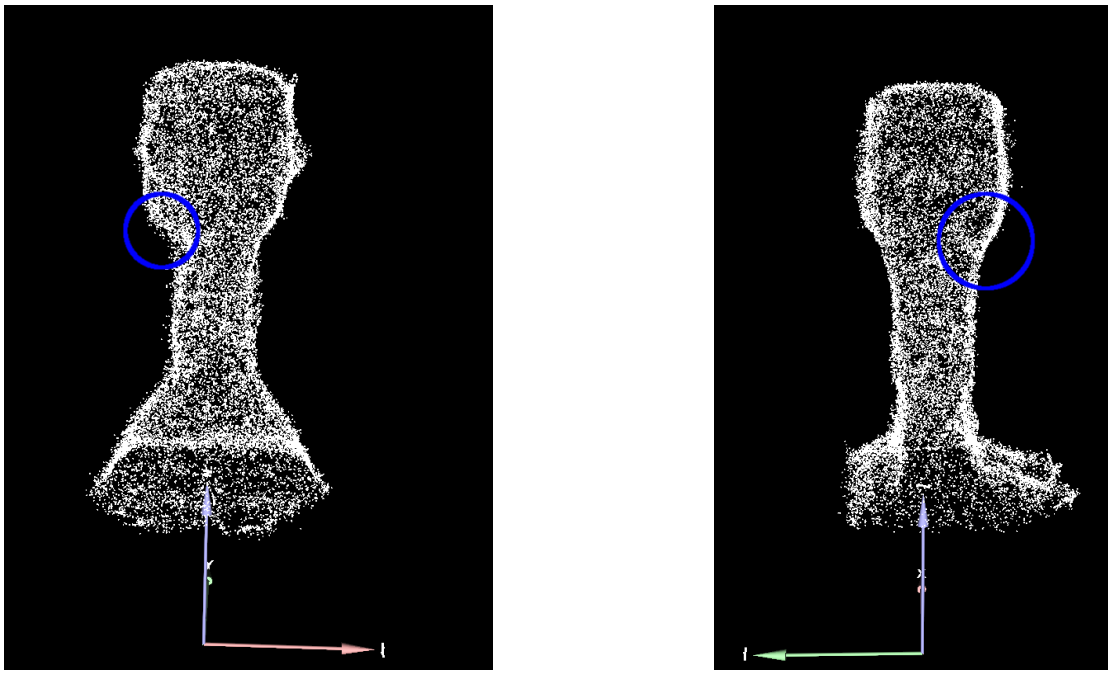

Figure 3.21: The original filter with double noise radius, blue ellipse illustrate the blurring of an edge

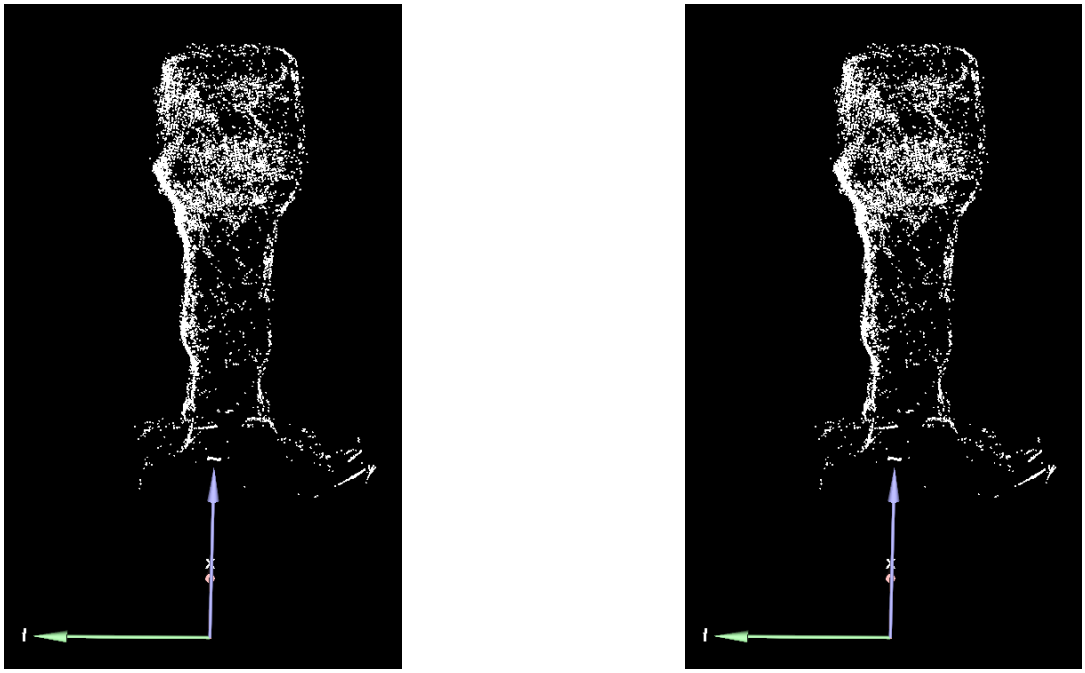

Figure 3.22: The original filter with a too small reduction radius leads to holes in the scan data

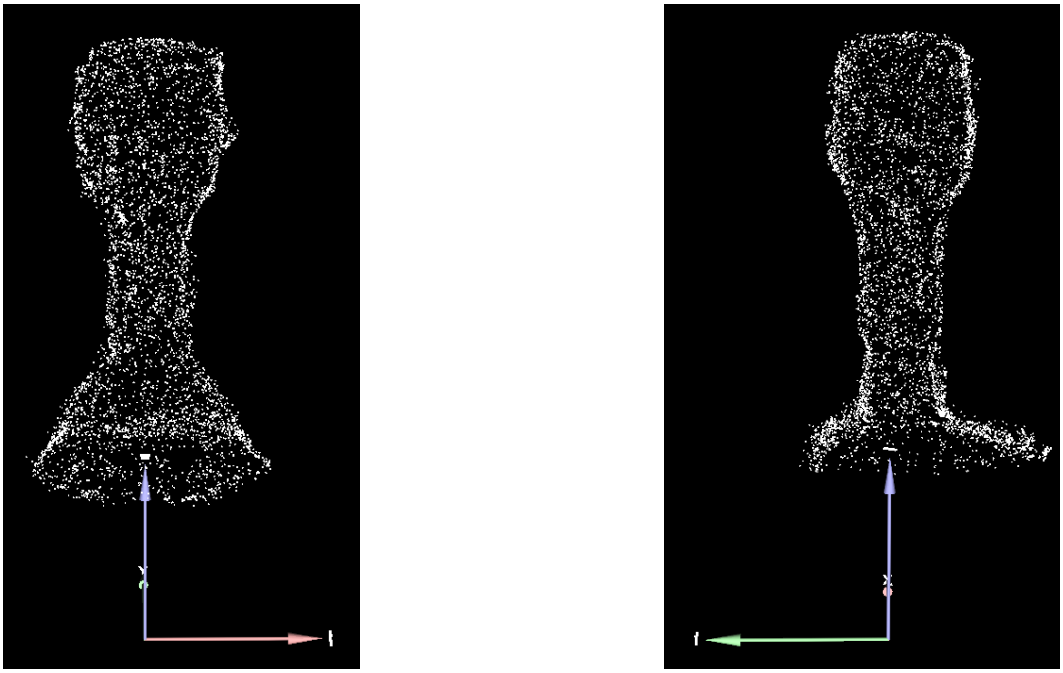

Figure 3.23: The original filter with double reduction radius

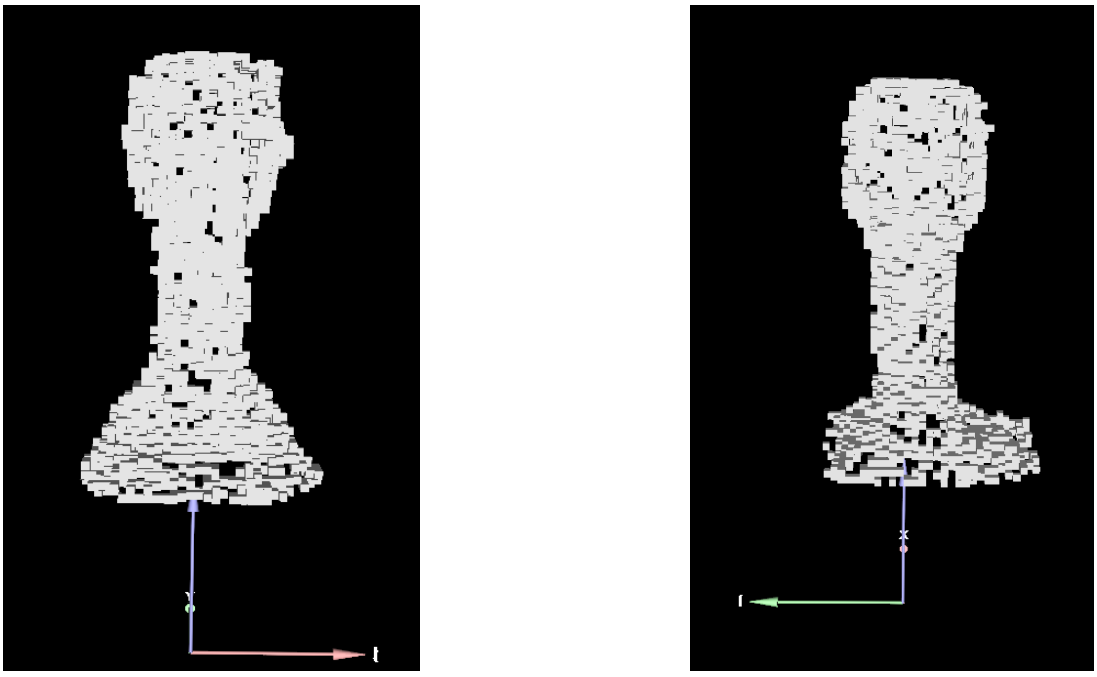

Figure 3.24: The resulting surface space for the inflating stage produced by double reduction radius

3.4 Surface Reconstruction

After registrating and filtering the scanned data, the inflating stage is performed on both objects. The results can be seen in fig. 3.25 and fig. 3.26. The mesh illustrates the surface of the real objects roughly and can be further processed.

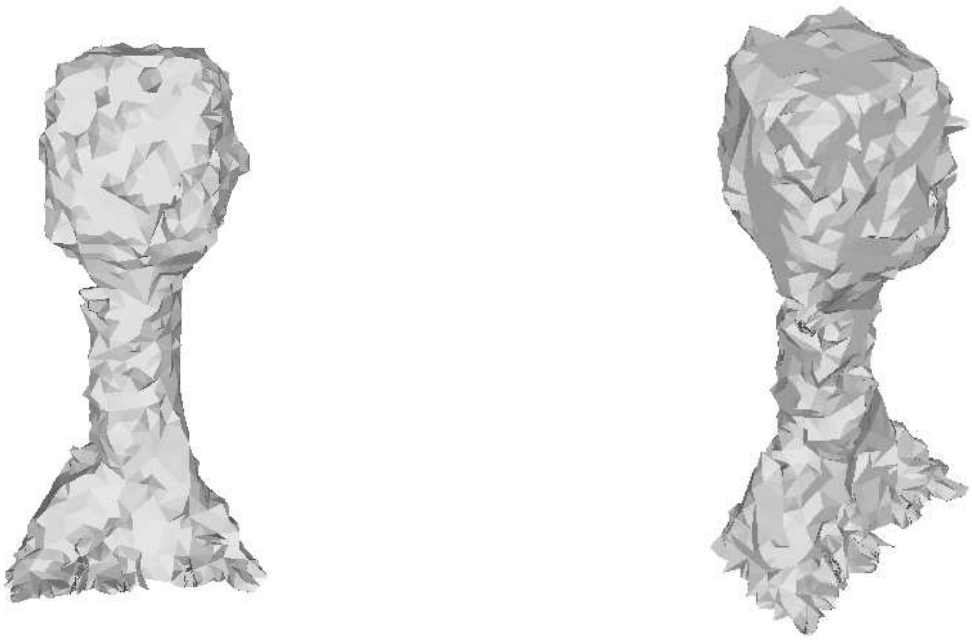

Figure 3.25: The mesh of the filter object after the inflating stage

For preparing the mesh for the detailing stage, we use the smoothing and regularizing techniques and get a better formed model that can be seen in fig. 3.27 and fig. 3.28. The surfaces are now a lot smoother and we have gained more regular triangles.
The detailing stages tries to reconstruct more details from the scanned data. It results in meshes as fig. 3.29 and 3.30.
Given the impreciseness and noisiness of both pose and depth measurement, the surface reconstruction algorithm is able to build a raw mesh model of the presented objects. These models have the advantage of being watertight and therefore, representing the surface of a real object in a more practical way.
In fig. 3.31 and fig. 3.32 both, the manually made models and the automatic models using the omniRob system are each shown in a single image. The automatically objects can only give a rough estimation of the actual surface. The alignment of the models was performed using the aforementioned ICP algorithm. The Coordinate Root Mean Squared Error (cRMS) is the average point-to-point distance of points to their correspondences after alignment and is shown for both objects in tab. 3.4.

Object	Correspondences	cRMS [mm]
Filter	24108	2.7963
Coffee	18312	2.4753

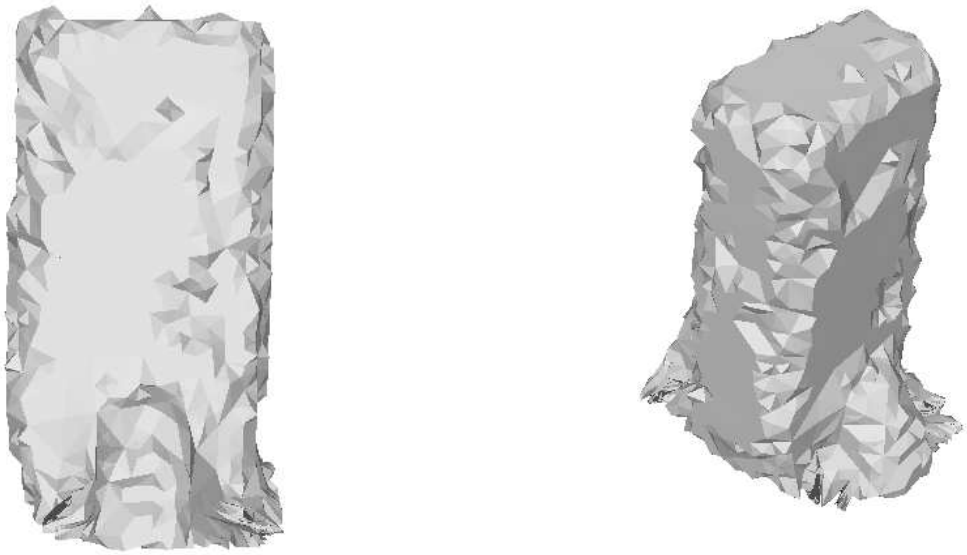

Figure 3.26: The mesh of the coffee package after the inflating stage

Please note that in case of the models created using the omniRob platform, part of the grab jaws are modelled as well, refer to the left picture of fig. 3.31 and fig. 3.32. Therefore, for not falsifying the result, we cut off those parts of the model. Visible in the right picture of fig. 3.31 and fig. 3.32, for both objects some surfaces are clearly displaced. Furthermore, please note that for the coffee package, the automatically scanned mesh is thicker than the original object and therefore displaced after alignment. For those reasons, the registration results in an cRMS of $\sim$ 2.8mm for the filter object and $\sim$ 2.5mm for the coffee package.
As utilizing hardware that produce highly noisy measurements prevents us from estimating the accuracy of our approach, the next section analyses this aspect using a different hardware setup.

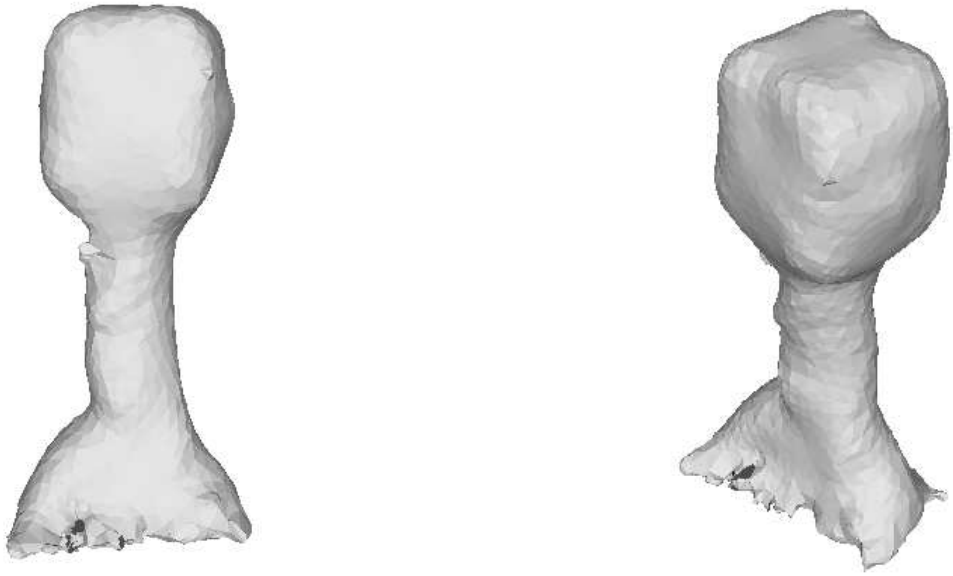

Figure 3.27: The mesh of the filter object before the detailing stage

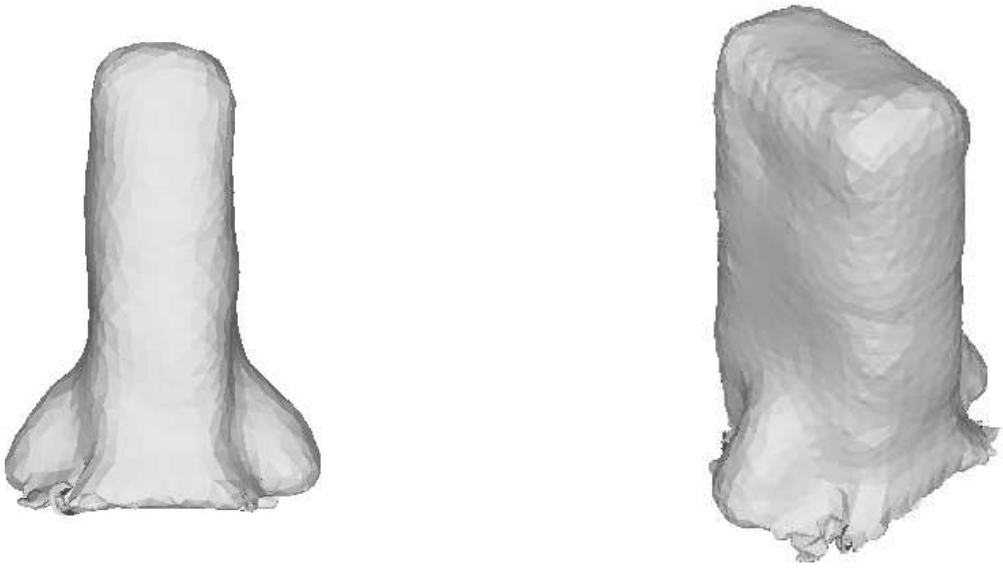

Figure 3.28: The mesh of the coffee package before the detailing stage

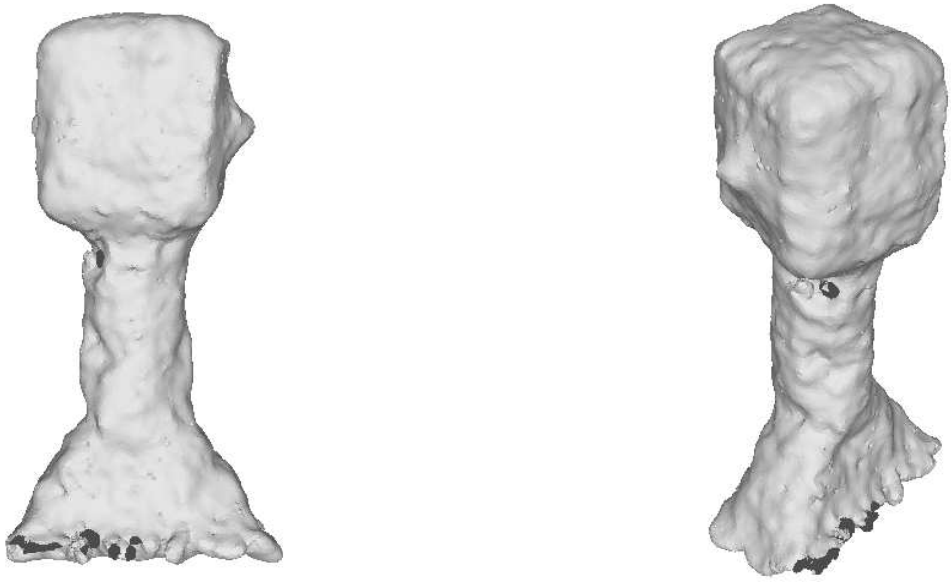

Figure 3.29: The final mesh of the filter object

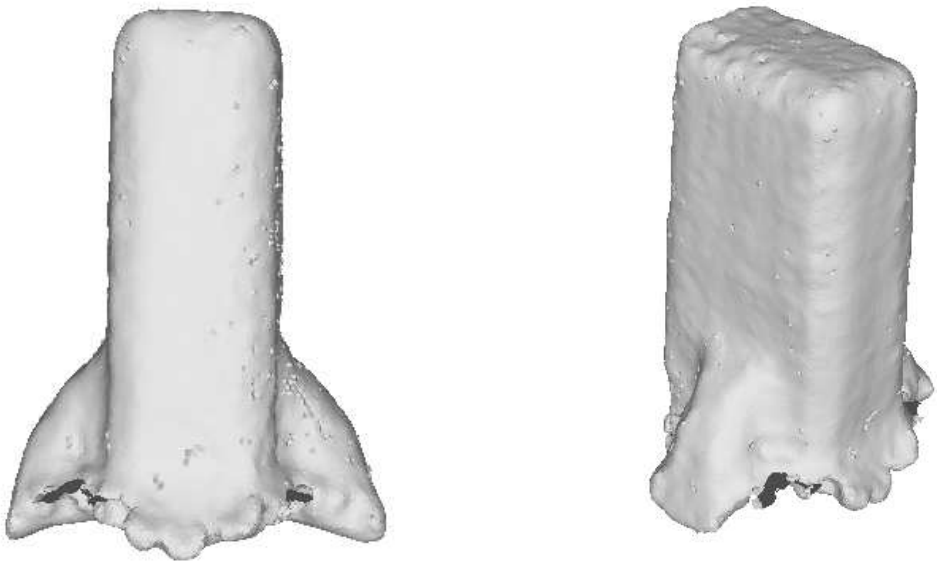

Figure 3.30: The final mesh of the coffee package

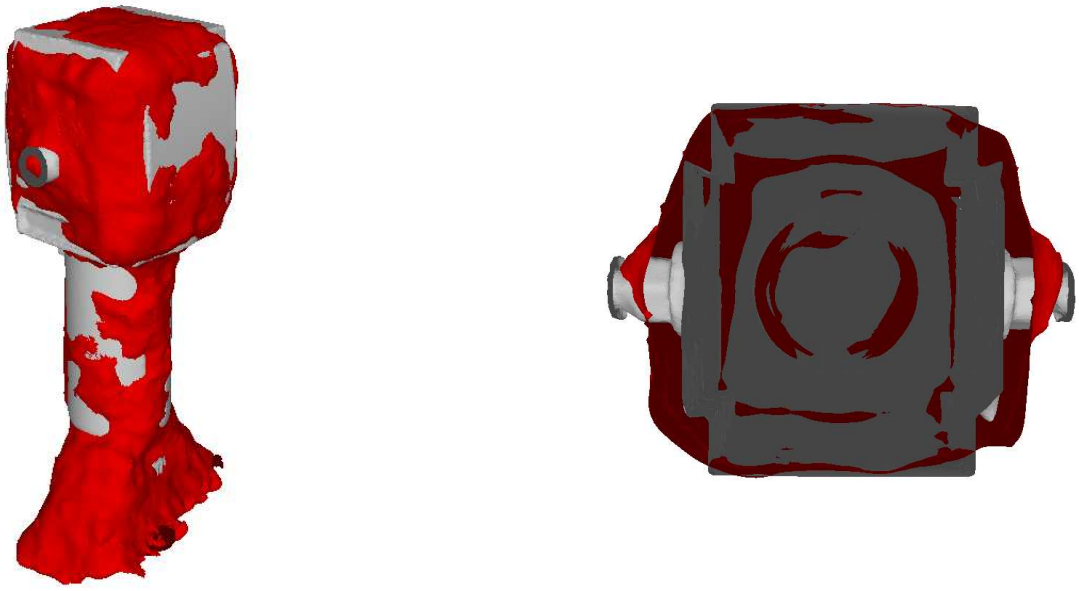

Figure 3.31: Comparison of the manually (white) and automatically (red) made models for the filter object

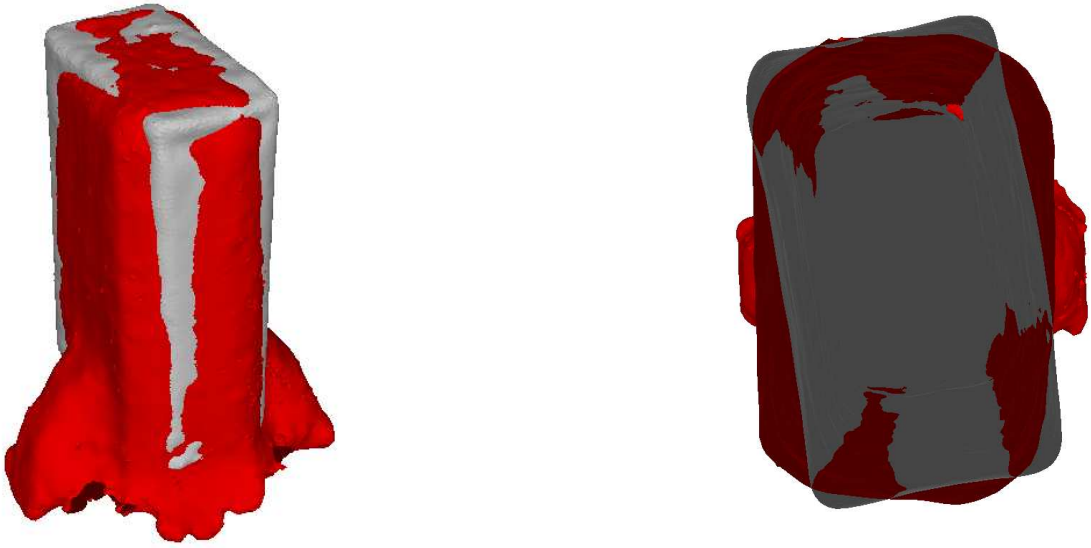

Figure 3.32: Comparison of the manually (white) and automatically (red) made models for the coffee package

3.5 Laser Scanned Objects

For comparison we used a different experimental setup consisting of a KUKA KR16-2 manipulator, see [41], with a ScanControl 2700 − 100 laser striper from Micro-Epsilon [42] mounted on it. Both devices are significantly more accurate in comparison to the KUKA LWR4+ and the used stereo-system. Using our surface reconstruction algorithm combined with the new hardware on the filter object and the coffee package, the created mesh models show a highly improved result as can be seen in fig. 3.33 and fig. 3.34.

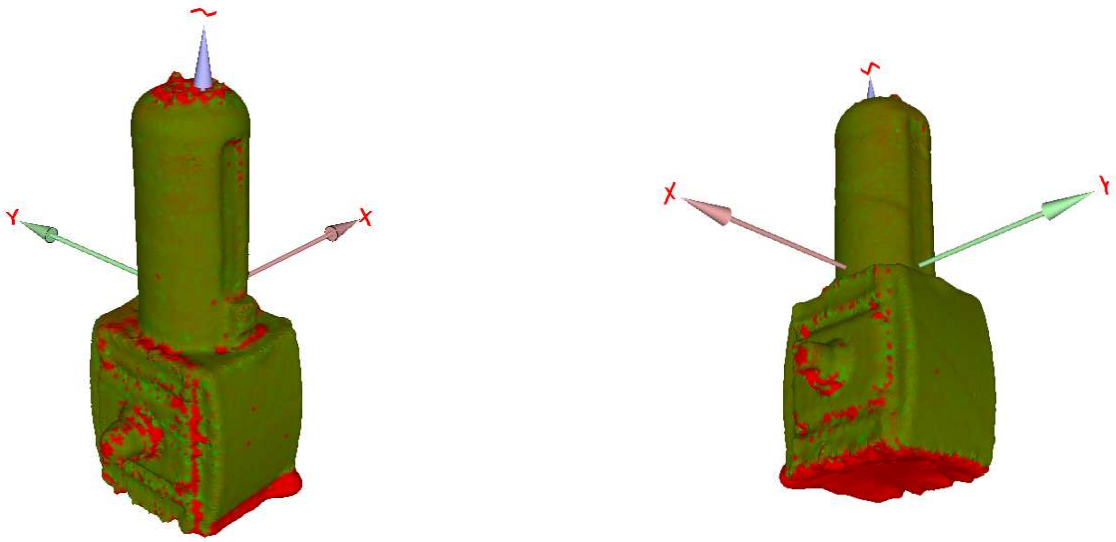

Figure 3.33: The final mesh of the filter object using a laser striper

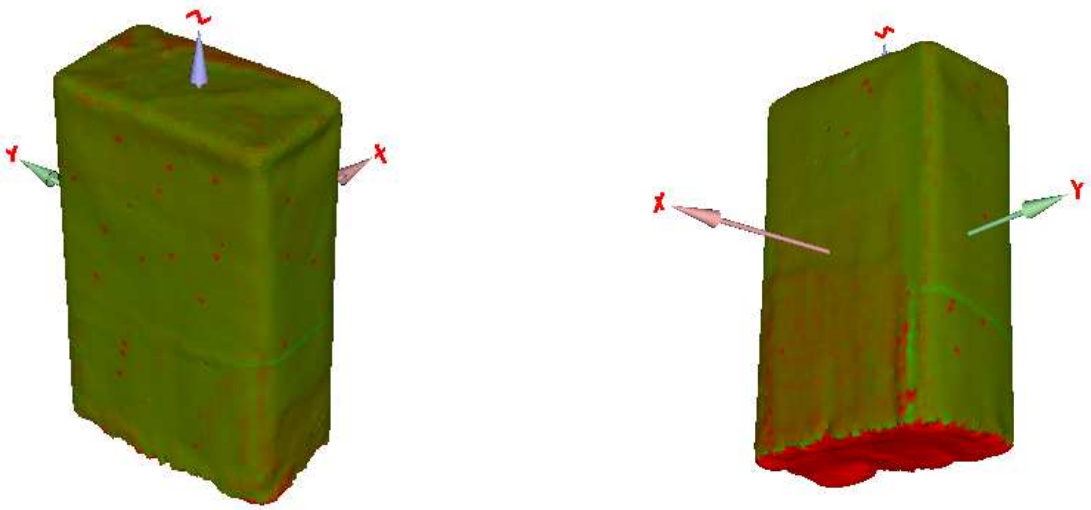

Figure 3.34: The final mesh of the coffee package using a laser striper

Please note, that for the filter object the top cylinder is missing. This is due to the small connection between the body and this cylinder. Another reason is that there are only few noisy measurements of this connection and consequently, the mesh cannot inflate through this area.
For both objects, we mapped the estimated variance of the distance from vertex to nearby measurements as colour to the triangles. Here, red illustrates a large variance or even no measurements at all while green represents a small variance. This can be interpreted as confidence of the position of the triangles and their vertices.
In fig. 3.35 and fig. 3.36, the manually scanned and the automatically build model are illustrated together. The alignment for both was computationally done with the aforementioned ICP algorithm. Its result gives a hint on the preciseness of the model and can be seen in table 3.5. Here, the cRMS is for the coffee package under 0.7mm. For the filter object, a larger error occurs as on the one hand the top cylinder is not modelled and on the other hand, a number of high variance parts show that some details of the object were not scanned.

Object	Correspondences	cRMS [mm]
Filter	26397	1.5590
Coffee	22491	0.6875

Nevertheless, for both objects we can state that our surface reconstruction approach is able to recover most details and realistically reconstruct the scanned object.

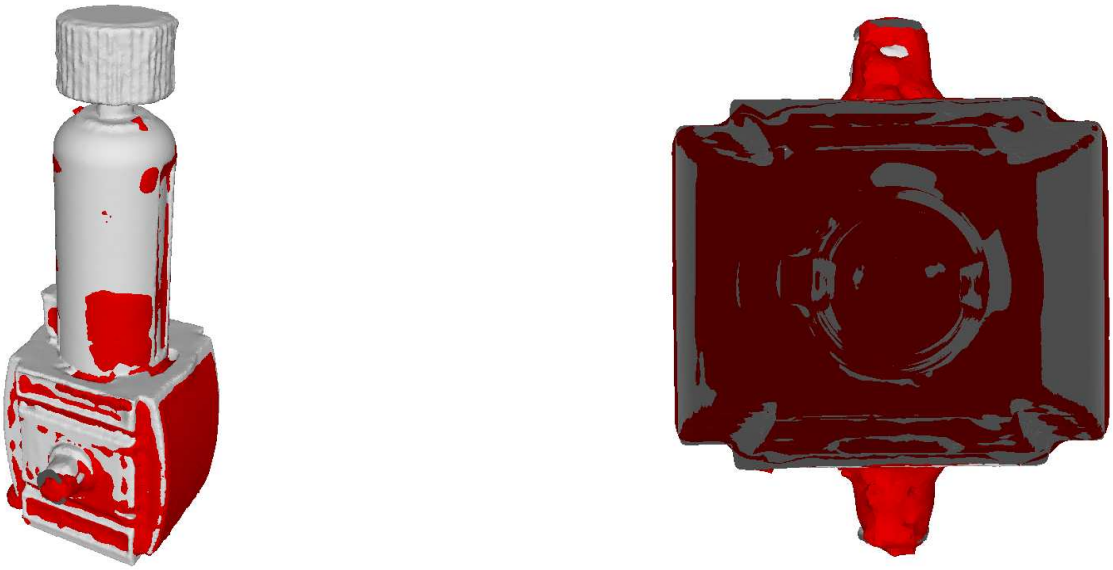

Figure 3.35: Comparison of the manually (white) and automatically (red) made models for the filter object

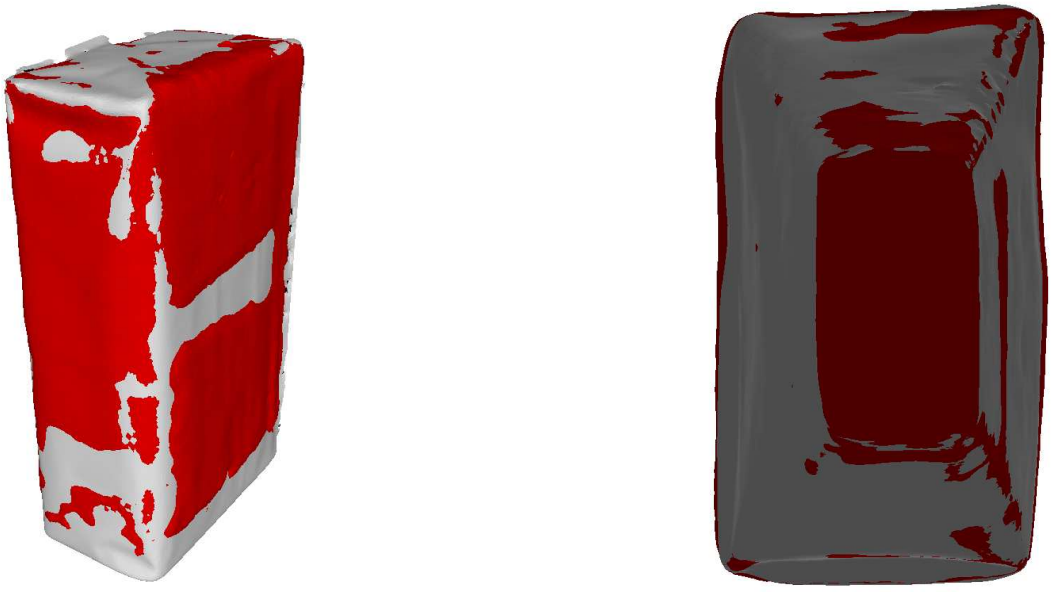

Figure 3.36: Comparison of the manually (white) and automatically (red) made models for the coffee package

Chapter 4

Conclusion and Outlook

In this work, we implemented a method for surface reconstruction. For acquiring the sensor data, different poses have to be planned. Therefore, we implemented a NBV planning algorithm, that allows for automatic data acquisition. This is done via estimating the space inside the to be scanned object. From this space, a point is projected through the surface or possible borders with a user defined distance, to obtain the position of a viewpoint candidate. The information gain for those viewpoint candidates is then estimated and the highest information gain is chosen as next viewpoint.
Once the data is gained, we have to cope with pose errors. Therefore, we use the iterative closest point (ICP) algorithm. While in most cases, the ICP algorithm reliably recovers the pose of the measurement, some cases remain that compromise the surface reconstruction. We developed a filtering stage in order to filter those and outliers for improving the measurement confidence. By comparing the estimated normals of nearby points and combining similar measurements, an estimation for the real surface can be calculated.
Through inflating, we can build a raw mesh model of the scanned object. For this, a further space needs to be created that represents the inside space of the mesh. By projecting from this space through the single vertices, the expanding direction can be determined. For avoiding growing through holes in the measurements, we first cluster the measurements in a voxelspace. A safe movement of each vertex can be determined by searching in expanding direction for a voxel with measurements attached to it. Through this, vertices that grow through holes in the surface space can be avoided.
By applying smoothing and regularizing techniques that highly depend on the connected topology of a watertight mesh, we prepare the model for the detailing stage in order to gain regular triangles with a smooth surface. In this stage, we try to recover finer details of the model by moving vertices to nearby measurement points. We show experimentally that facing imprecise and noisy data, a coarse model can be recovered. Even though this model is watertight, but faces high noise using the setup of omniRob platform. By using preciser hardware, a surface model with fine

details can be build. All mesh models are watertight, meaning there are no holes in the mesh. This is achieved through a deformable mesh with topology adaptation. Those meshes can be used for object recognition or planning of manipulation task. Based on this, pick and place tasks can be implemented as gripping and placing of an object can be planed using the mesh model. Consequently, having a mesh representation of an object, is one step towards enabling robots to assist humans.
Through calculating the distance variance from vertex to measurements, poorly scanned areas of the model can be detected. Consequently, in future work a post modelling scan process can improve details of a mesh model by measuring those areas. As the mesh is deformable, those new scans can directly push the corresponding triangles to the actual surface.
Another improvement of our approach might to cluster the scanned data and perform a surface reconstruction on the acquired clusters. For our filter object this would mean to separate the scanned data for the body and the top cylinder. For both clusters a mesh model needs to be build. Based on this, merging techniques need to be developed in order to combine several meshed parts of an object without loosing the topology of the mesh.
Further development of this approach might include performance optimization. Therefore, parallel processing is one important key feature. One approach to achieve that is to parallelize the stages of our approach. As only a voxel based surface representation with as few as possible holes is necessary for the inflating stage, the scanning phase of the algorithm can be divided into an exploring stage and a fine measuring phase. For the first phase, only a small amount of measurements would be necessary to cover all surfaces of an object. This step can be achieved by manipulating the evaluation parameters of our NBV approach. Furthermore, lowering the minimum number of measurements per voxel to 1, the needed space could be generated quicker. After reaching a certain surface coverage, the inflating stage could be started in parallel to the fine measuring phase. In the latter, the NBV algorithm is adjusted to the original parameters, for increasing measurement confidence.
After the inflating stage finishes, the detailing stage can be initiated and the new gained scans can be directly forwarded to it. This could result in a large performance improvement.
A second attempt for performance optimisation is to parallelize the single steps of the algorithms. To a large extent, many calculations are performed individually for each point or vertex. Making use of the concept of General Purpose Computation on Graphics Processing Unit (GPGPU) can largely reduce processing time by parallelizing those calculations.
A further improvement could consist of a surface reconstruction that performs directly after the first scan. This can be achieved through adapting the inflating stage to perform on a not closed voxel surface representation. Therefore, we need to enable safe movements to influence the movements of neighbouring and connected vertices. Consequently, a smoother inflating of a mesh can be obtained and the impact of large holes can be significantly reduced. Based on this, a parallel scanning

and meshing approach can be implemented.

List of Figures

List of Abbreviations

NBV Next Best View 6
cRMS Coordinate Root Mean Squared Error 47
ICP Iterative Closest Point 6
MVS Mass Vector Sum 6
PTU Pan Tilt Unit 31
GPGPU General Purpose Computation on Graphics Processing Unit 56

Bibliography

[1] D. Lamb, D. Baird, and M. A. Greenspan, "An automation system for industrial 3-d laser digitizing," in *1999. Proceedings. Second International Conference on 3-D Digital Imaging and Modeling*, 1999.

[2] S. Khalfaoui, R. Seulin, Y. Fougerolle, and D. Fofi, "View planning approach for automatic 3d digitization of unknown objects," in *Computer Vision ECCV 2012. Workshops and Demonstrations*. Springer Berlin Heidelberg, 2012.

[3] B. Loriot, R. Seulin, and P. Gorria, "Non-model based method for an automation of 3d acquisition and post-processing," *Electronics Letters on Computer Vision and Analysis*, 2008.

[4] N. A. Massios and R. B. Fisher, "A best next view selection algorithm incorporating a quality criterion," in *BMVC*, 1998.

[5] S. Kriegel, T. Bodenmüller, M. Suppa, and G. Hirzinger, "A surface-based next-best-view approach for automated 3d model completion of unknown objects," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[6] S. Kriegel, C. Rink, T. Bodenmüller, A. Narr, M. Suppa, and G. Hirzinger, "Next-best-scan planning for autonomous 3d modeling," in *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, 2012.

[7] S. Kriegel, C. Rink, T. Bodenmüller, and M. Suppa, "Efficient next-best-scan planning for autonomous 3d surface reconstruction of unknown objects," *Journal of Real-Time Image Processing*, pp. 1–21, 2013.

[8] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992.

[9] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Februrary 1992.

[10] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, 2001.

[11] R. Toldo, A. Beinat, and F. Crosilla, "Global registration of multiple point clouds embedding the generalized procrustes analysis into an icp framework," in *3DPVT 2010 Conference*, 2010.

[12] T. Masuda, K. Sakaue, and N. Yokoya, "Registration and integration of multiple range images for 3-d model construction," in *Proceedings of the 13th International Conference on Pattern Recognition*, 1996.

[13] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal for Computer Vision*, vol. 13, pp. 119–152, 1994.

[14] P. J. Neugebauer, "Geometrical cloning of 3d objects via simultaneous registration of multiple range images," in *1997 International Conference on Shape Modeling and Applications*, 1997.

[15] D. Chetverikov, D. Svirko, D. Stepanov, and P. Drsek, "The trimmed iterative closest point algorithm," in *International Conference on Pattern Recognition*, 2002, pp. 545–548.

[16] D. A. Simon, "Fast and accurate shape-based registration," Ph.D. dissertation, Carnegie Mellon University, 1996.

[17] J. Zhu, N. Zheng, Z. Yuan, and S. Du, "Point-to-line metric based iterative closest point with bounded scale," in *ICIEA 2009. 4th IEEE Conference on Industrial Electronics and Applications*, 2009.

[18] M. Potmesil, "Generating models of solid objects by matching 3d surface segments," in *IJCAI*, 1983.

[19] M. Suppa, "Autonomous robot work cell exploration using multisensory eye-in-hand systems," Ph.D. dissertation, Gottfried Wilhelm Leibniz Universit"at Hannover, 2008.

[20] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking," in *2012 Second International Conference on3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, 2012.

[21] F. Pomerleau, A. Breitenmoser, M. Liu, F. Colas, and R. Siegwart, "Noise characterization of depth sensors for surface inspections," in *2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI)*, 2012.

[22] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," in *Robotics: Science and Systems*, 2009.

[23] G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, 1994.

[24] G. Pandey, J. McBride, S. Savarese, and R. M. Eustice, "Visually bootstrapped generalized icp," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[25] K. Pulli, "Multiview registration for large data sets," in *1999. Proceedings. Second International Conference on 3-D Digital Imaging and Modeling*, 1999.

[26] S. Krishnan, P. Y. Lee, J. B. Moore, and S. Venkatasubramanian, "Global registration of multiple 3d point sets via optimization-on-a-manifold," in *Eurographics Symposium on Geometry Processing*, 2005, pp. 187–196.

[27] N. J. Mitra, N. Gelfand, H. Pottmann, and L. Guibas, "Registration of point cloud data from a geometric optimization perspective," in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM, 2004, pp. 22–31.

[28] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International journal of computer vision*, vol. 1, no. 4, pp. 321–331, 1988.

[29] J. V. Miller, D. E. Breen, W. E. Lorensen, R. M. O'Bara, and M. J. Wozny, "Geometrically deformed models: a method for extracting closed geometric models form volume data," in *ACM SIGGRAPH Computer Graphics*, vol. 25, no. 4. ACM, 1991, pp. 217–226.

[30] M. Vasilescu and D. Terzopoulos, "Adaptive meshes and shells: Irregular triangulation, discontinuities, and hierarchical subdivision," in *In Proceedings of Computer Vision and Pattern Recognition conference*, 1992.

[31] J.-O. Lachaud and A. Montanvert, "Deformable meshes with automated topology changes for coarse-to-fine three-dimensional surface extraction," *Medical Image Analysis*, vol. 3, no. 2, pp. 187 – 207, 1999.

[32] J.-Y. Park, T. McInerney, D. Terzopoulos, and M.-H. Kim, "A non-self-intersecting adaptive deformable surface for complex boundary extraction from volumetric images," *Computers & Graphics*, vol. 25, no. 3, pp. 421–440, 2001.

[33] Y. Chen and G. Medioni, "Description of complex objects from multiple range images using an inflating balloon model," *Computer Vision and Image Understanding*, vol. 61, no. 3, pp. 325–334, 1995.

[34] A. Lachaud, Jacques-Olivier; Montanvert, "Deformable meshes with automated topology changes for coarse-to-fine three-dimensional surface extraction," *Medical Image Analysis*, 1999.

[35] O. Monga and R. Deriche, "3d edge detection using recursive filtering: application to scanner images," in *Computer Vision and Pattern Recognition, 1989. Proceedings CVPR '89., IEEE Computer Society Conference on*, Jun 1989, pp. 28–35.

[36] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[37] D. Jung and K. K. Gupta, "Octree-based hierarchical distance maps for collision detection," in *1996 IEEE International Conference onRobotics and Automation, 1996. Proceedings.*, 1996.

[38] (2014, November). [Online]. Available: http://www.kuka-labs.com/de/service_robotics/mobile_robotics/omniRob/

[39] (2014, November). [Online]. Available: http://www.alliedvisiontec.com/us/products/cameras/firewire/guppy-pro/f-125bc.html

[40] (2014, November). [Online]. Available: http://www.kuka-labs.com/en

[41] (2014, November). [Online]. Available: http://www.kuka-robotics.com/germany/en/products/industrial_robots/low/kr16_2/start.htm

[42] (2014, November). [Online]. Available: http://www.micro-epsilon.de/laser-scanner/scanCONTROL/Laser-scanner-selection/index.html

License